UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

CONTROL EN LAZO CERRADO MEDIANTE FPGA CON ADC INTEGRADO

Pablo Amor Peinado
Tutor: Ángel de Castro Martín

JUNIO 2018

CONTROL EN LAZO CERRADO MEDIANTE FPGA CON ADC INTEGRADO

AUTOR: Pablo Amor Peinado
TUTOR: Ángel de Castro Martín

HCTLab
Dpto. de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2018

Resumen

Este Trabajo de Fin de Grado tiene como objetivo principal la implementación de control digital en una FPGA de bajo coste con ADC embebido.

Para ello se ha escogido una FPGA de la familia MAX10 de Altera con ADC integrado, que a día de hoy no se emplea habitualmente para dicha función, por lo que primero se ha tenido que proceder a la familiarización por parte del alumno con estas tecnologías.

A continuación, se ha pasado a diseñar un regulador apto para realizar el control en lazo cerrado del convertidor conmutado escogido, que en este caso ha sido un reductor o *Buck* multifase. Este primer diseño teórico se ha hecho para el caso más inestable de todos para que así el regulador sea capaz de controlar todos los casos sin problema y se ha visto la importancia de que prime la estabilidad sobre la rapidez en este tipo de sistemas.

Una vez que se ha obtenido el regulador deseado, tras ajustar la ganancia para que su actuación no supere en ningún momento el valor en régimen permanente, se ha pasado a su implementación digital en VHDL, donde se ha visto el gran impacto que tiene una correcta elección de la resolución de los coeficientes del regulador para lograr la respuesta esperada.

Después de obtener resultados tanto de tensión como de corriente mediante simulaciones para el caso de una fase y el de cuatro fases, se ha llevado a cabo el montaje hardware del circuito propuesto en el laboratorio HCTLab de la EPS con el objetivo de obtener resultados más cercanos a la realidad y realizar pues la comparativa entre resultados teóricos y prácticos.

Palabras clave

Convertidor CC/CC, reductor multifase, control en lazo cerrado, estabilidad, resolución, FPGA, control digital

Abstract

The main objective of this Bachelor's Thesis is the implementation of digital control in a low cost FPGA with embedded ADC.

For that purpose an FPGA from Altera's MAX10 family with an integrated ADC was chosen. These FPGAs are not normally used for this type of function and that is the reason why, first of all, the student had to get used to these technologies.

Then, a regulator capable of controlling the selected switched converter, in this case a multiphase Buck converter, was designed. This first design was made for the least stable case so that the regulator could be able to control the Buck converter in any of the situations. Also, the importance of stability versus speed in these systems was made clear.

Once the desired regulator was obtained and after having to adjust its gain a bit so the actuation never exceeded at any point the value of the desired voltage, the design was digitally implemented in VHDL, where the great impact of a proper resolution of the regulator coefficients to obtain the expected response was observed.

After obtaining both voltage and current results for one and four phases by simulations, the designed circuit was assembled in hardware in the HCTLab of the EPS to be able to obtain more grounded and real results and that way compare the ideal software results and the hardware ones.

Keywords

DC/DC Converter, multiphase Buck converter, closed loop control, stability, resolution, FPGA, digital control

Agradecimientos

Me gustaría dar las gracias en primer lugar a Ángel, tanto por sus valiosas clases como por su apoyo y tutela durante este último año. Gracias por tu dedicación y sobre todo por la confianza que has depositado en mí para llevar este trabajo a buen puerto.

Gracias también a mis compañeros, que durante esta carrera nos hemos apoyado mutuamente para superar los retos más insospechados. En particular Jorge, Nacho y Jiayu.

También agradezco a mis padres por poner los medios necesarios para superar este grado a mi alcance y por depositar toda su confianza en mí. ¡Ya tenéis otro ingeniero!

A continuación, me gustaría dar las gracias a mi hermano Álvaro. Siempre estás a mi lado, tanto en lo bueno como en lo malo. Te deseo de corazón que la vida te depare todo lo mejor. De verdad que no sé qué haría sin ti.

Por último, que no menos importante, me gustaría dar las gracias a mi director de cine favorito: Sion Sono. Gracias a ti y a otros tantos autores, que he descubierto a lo largo de la carrera, por mostrarme la luz en los momentos más oscuros y ayudarme durante este periplo.

Pablo Amor Peinado

Junio 2018

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

1.1 Motivación

Este Trabajo de Fin de Grado se centra en la implementación de un controlador digital para un convertidor *Buck* multifase implementado en una *Field Programmable Gate Array* (FPGA) MAX10 de Altera [1]. El conversor analógico digital (ADC) integrado de esta FPGA, junto con su memoria no volátil, permite el funcionamiento de esta FPGA sin necesidad de otros componentes externos con alguna excepción, como bien puede ser un reloj. Esto abre nuevas posibilidades para la implementación de reguladores digitales, ya que una FPGA de bajo coste puede ser usada sin prácticamente ningún componente externo para realizar control en lazo cerrado.

Las FPGAs de estas características, que incorporan el ADC, son relativamente novedosas y, por ello, aún no han sido empleadas de forma masiva para el control en lazo cerrado.

Así pues, la motivación principal de este TFG consiste en la elaboración de un diseño de control clásico de tipo PID, pero con la dificultad de enfrentarse a tecnologías recientes y poco exploradas, al menos para dicha aplicación, conllevando por ello una iniciación en el campo de la investigación.

Otro punto de interés consiste en el empleo como planta en el diseño de un convertidor conmutado, en este caso un reductor. Este tipo de convertidores se emplean, de forma muy frecuente a día de hoy, como fuentes de alimentación en múltiples sistemas electrónicos y, por ello, resulta de gran importancia conocer su funcionamiento.

Durante la implementación del regulador se optará por el empleo de notación en coma fija, bastante susceptible a problemas de resolución, y se tratará de encontrar un método sencillo para estimar aproximadamente la resolución a escoger a través de los coeficientes del regulador.

Además, se realizará un montaje hardware del diseño realizado para estudiar la comparación entre teoría y práctica a través de una gran cantidad de simulaciones y medidas.

1.2 Objetivos

Si bien es cierto que en este trabajo se partía de una serie de objetivos concretos, otros tantos han ido surgiendo de forma dinámica durante la realización del mismo ante la aparición de nuevos retos y problemas a resolver. Si se contabilizan de forma global la lista de objetivos final quedaría de la siguiente forma:

1.) Familiarización con FPGA MAX10 de Altera con ADC integrado.

2.) Estudio del empleo de dicho modelo de FPGA para control en lazo cerrado.

3.) Diseño teórico de un regulador para el control de un convertidor *Buck* multifase.

4.) Observación de las consecuencias que produce intentar diseñar el regulador más rápido posible frente a uno más estable.

5.) Implementación de un sistema en lenguaje VHDL para el control digital.

6.) Análisis del impacto de la resolución en coma fija de los coeficientes del regulador en la dinámica del conjunto.

7.) Montaje hardware del circuito diseñado y toma de medidas para comparación teoría/práctica.

1.3 Organización de la memoria

Esta memoria se encuentra organizada de la siguiente forma:

- En el capítulo 2 se exponen las características del modelo de la FPGA escogida, así como la configuración del ADC por la que se ha optado para realizar el diseño. Además, se justifica la elección de un convertidor reductor para su control.

- En el capítulo 3 se procede, por un lado, al diseño teórico del regulador y se comentan los problemas de saturación surgidos al plantear un sistema muy rápido. Por otro lado, se lleva a cabo la implementación en VHDL de los distintos bloques que componen el circuito de control, se explica el *timing* de los mismos y, por último, se realiza un análisis del impacto de la resolución de los coeficientes del regulador en el comportamiento del sistema.

- En el capítulo 4 se explica cómo se ha realizado el montaje hardware y se toman medidas del mismo tanto para el caso de una fase como para el de cuatro fases, llevando a cabo al final una comparación entre éstos y los extraídos mediante simulación software.

- Por último, el capítulo 5 está dedicado a la extracción de conclusiones del trabajo presentado y al planteamiento de las líneas de estudio o investigación futuras que se podrían llevar a cabo tras su finalización.

2 Estado del arte

2.1 FPGA

Las FPGAs cuentan a día de hoy con una gran variedad de aplicaciones, debidas principalmente a su versatilidad y a la flexibilidad que ofrecen. Estas facilidades hacen que numerosos diseñadores opten por incorporar en sus proyectos este tipo de tecnología y, por ello, se trata de un campo actualmente en expansión. De todas estas aplicaciones si hubiera que destacar una sería el procesamiento digital de señales (DSP), debido al alto nivel de paralelismo que ofrecen que se traduce en una gran potencia computacional y en una alta eficiencia para estas operaciones. Si hay una contrapartida que los diseñadores de sistemas digitales ven en las FPGAs es el empleo de lenguajes como VHDL, en los que hay que declarar lo que sucede en cada uno de los ciclos de reloj de forma explícita, prefiriendo (o estando más familiarizados con) lenguajes de alto nivel como C. Esta cuestión se puede resolver mediante el empleo de varias técnicas, como bien pueden ser los IP Cores o programas que traducen código en lenguaje de alto nivel a lenguaje VHDL.

Por supuesto, una de las aplicaciones que se le suele dar a una FPGA en el campo del tratamiento de señales no es otro que el que es objeto de estudio en este TFG: el control en lazo cerrado. Este tipo de control es propio de sistemas que se encuentran realimentados, es decir, donde la entrada es función de la salida. En el panorama actual hay múltiples ejemplos de estos sistemas: termostatos, hornos, reguladores de voltaje o corriente, etc.

La FPGA escogida para la realización de dicho control en lazo cerrado fue una de la familia MAX10 de Altera [2], en particular el modelo: MAX10 10M50DAF484C7G [3]. Dicha FPGA viene montada en la placa DE10-Lite de Terasic [4]. Entre las prestaciones principales que presenta esta placa destacan: un acelerómetro, conectores para Arduino, cuarenta pines GPIO, una conexión para alimentación a 5 V y para USB, una SDRAM de 64 MB, un conector VGA de cuatro bits, seis displays de siete segmentos, diez LEDs, diez interruptores (switches) y dos botones; además de, por supuesto, la citada FPGA.

La MAX10 10M50DAF484C7G es una *Field Programmable Gate Array* no volátil, que cuenta con un total de 50000 elementos lógicos, 3125 Logic Array Blocks, 360 I/Os, se encuentra montada por técnicas de montaje superficial SMD, es capaz de operar a temperaturas en el rango de 0 y 85 ºC y puede trabajar a frecuencias de hasta 450 MHz.

Lo común en este tipo de FPGAs es el empleo de conversores analógico/digital (ADC) externos. Para lograr la comunicación deseada entre ambos componentes bastaría con configurar las señales de la FPGA correctamente de manera que actuaran sobre las distintas señales del ADC.

Obviamente este proceso tarda un tiempo que disminuiría drásticamente si dicho conversor ADC se encontrara incorporado dentro de la FPGA, aumentando así las prestaciones. De todos modos, ésta no es la ventaja principal, ya que existen ADCs externos muy rápidos. Donde de verdad se nota es en el precio: soldar más chips y emplear una placa de mayor tamaño supone un coste adicional no despreciable, inexistente en caso de que el ADC venga embebido.

Se tratan de unas tecnologías bastante recientes y, por tanto, cuesta encontrar información incluso para realizar la configuración del ADC. Mucho más aún para realizar un control en lazo cerrado, función para la que aún no se ha popularizado el uso de dichas FPGAs.

A lo largo de este trabajo se investigará sobre la capacidad que tiene esta FPGA para realizar un control en lazo cerrado, un campo aparentemente poco explorado.

2.1.1 Configuración de ADC

El ADC se ha configurado mediante el software propietario de Intel, Quartus. Dicho programa ofrece una opción de integración de sistemas llamada Qsys System Integration Tool, que permite realizar el diseño del ADC que se va a emplear para más tarde generar el código correspondiente a dicho módulo de forma automática.

El diseño realizado [Figura 2-1] básicamente cuenta con un reloj a 50 MHz que se conecta a un lazo de seguimiento de fase (PLL) que, mediante un *clock bridge*, conecta dicho reloj al ADC (c0). Por otro lado, el ADC además necesita un reloj a 10 MHz tal y como figura en su respectiva hoja del fabricante [5], que también le proporciona el PLL (c1).

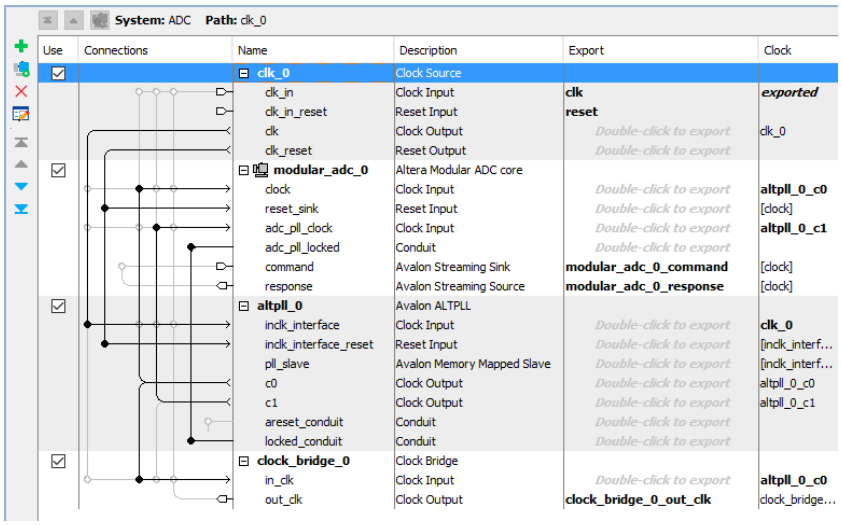


Figura 2-1: Diseño ADC

En cuanto a la configuración del ADC [Figura 2-2], se optó por emplear una variante del core del tipo "*ADC control core only*" que, si se observa la guía de usuario del ADC [5], se puede ver que se trata de la opción que ofrece más libertad para la gestión del ADC y, por ello, la más compleja de todas.
La fuente de tensión se eligió que fuera externa a un voltaje de 2,5 V y se empleó el canal 1 como entrada del ADC (el canal 0 está configurado por defecto como entrada analógica o Analog IN).
Tras seleccionar una frecuencia de muestreo de 1 MHz, se procedió a generar de forma automática el código necesario para la gestión del ADC.

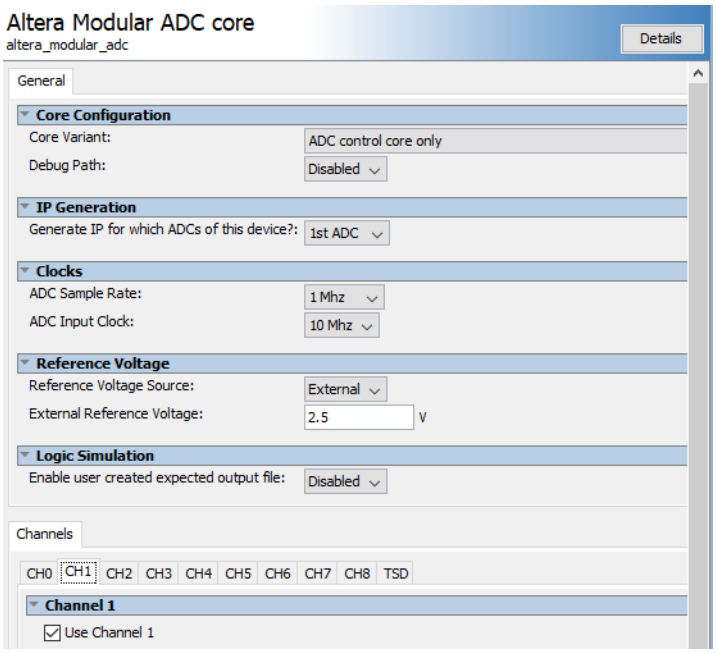


Figura 2-2: Configuración ADC

Para la gestión del ADC se optó por un sistema muy sencillo que únicamente comprueba en cada comento si el dato es válido y, en caso de que así sea, pone a la salida el valor leído por el ADC:

```
process (sys_clk)
begin
   if sys_clk'event and sys_clk = '1'  then
      if response_valid = '1' then
         adc_sample_data <= response_data;
         cur_adc_ch <= response_channel;
      end if;
   end if;
end process;
ADCout <= adc_sample_data;
```

Para más información de este código se recomienda visitar el Manual del programador [Anexo B].

2.2 Reductor

Todo sistema digital necesita una fuente de alimentación, por lo tanto, resulta de gran interés conocer sus principios de diseño.

Hay una gran cantidad de parámetros y factores a tener en cuenta para llevar a cabo el modelado de una de estas fuentes, como pueden ser: el voltaje (mayor o menor a la salida que a la entrada, CA o CC), la corriente (máxima y mínima), el ruido que se puede tolerar, la eficiencia (relación entre potencia disipada en la carga y potencia total), la necesidad de componentes externos o el asilamiento entre entrada y salida.

El esquema más básico de una fuente de alimentación [Figura 2-3] muestra cómo mediante un sistema de alimentación se obtiene a partir de un voltaje de entrada, un voltaje de salida determinado.

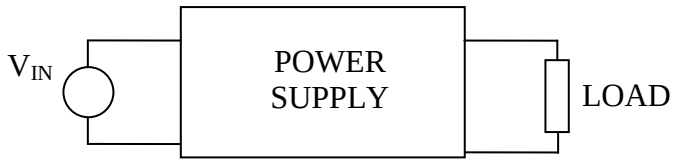


Figura 2-3: Fuente de alimentación

Sin embargo, lo que habitualmente se busca es la obtención de una tensión casi constante para regularla y que llegue lo más constante posible a la carga [Figura 2-4]. Para ello, tras pasar la tensión de entrada por un transformador, se rectifica haciéndola pasar por un puente de diodos que hace que la señal solo sea positiva. A continuación, la señal se suaviza mediante un filtro, que normalmente suele ser un condensador de gran capacidad y, por último, se pasa por un regulador que da una señal casi constante a la carga.

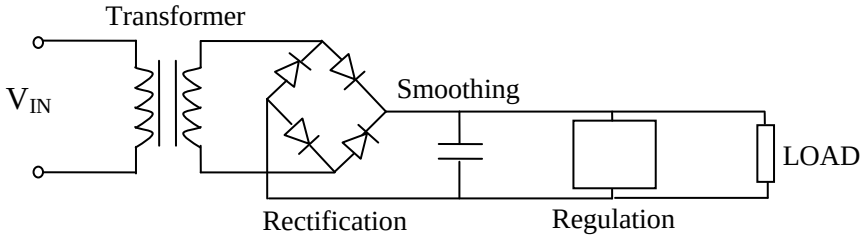


Figura 2-4: Fuente de alimentación regulada

En este trabajo se desea que a la salida haya un voltaje inferior que a la entrada (VIN>VOUT). Si bien se podría haber empleado un regulador lineal, debido a que cumple dicha regla de diseño y a que, por lo general, suponen un diseño bastante sencillo y con poco ruido (salida CC pura), suelen proporcionar una mala eficiencia.

Por ello, se ha optado por el empleo de un convertidor conmutado. Aunque son algo más ruidosos (salida no pura CC), este tipo de fuentes tienen una eficiencia mucho mayor, llegando fácilmente a superar 90% en la mayoría de los casos. De entre las distintas opciones que hay de este tipo de convertidores se ha optado por usar una de las topologías principales: *Buck* o *Boost*, y dado que se desea una tensión inferior a la salida que a la entrada, se ha elegido el modelo *Buck*, también llamado convertidor reductor.

En particular se ha optado por la versión multifase debido al menor condensador tanto a la entrada como a la salida que presenta, así como al mejor comportamiento ante altas corrientes. Además de permitir operar más allá de la corriente de saturación de un único inductor, lo que reduce el tamaño y el coste. En esta versión se cuenta con varios convertidores *Buck* actuando de forma paralela desfasados, cada uno de los cuales se denomina: fase.

3 Diseño del regulador

En este apartado se va a proceder a diseñar el regulador con el que se controlará el convertidor conmutado escogido, en este caso un reductor (o convertidor *Buck*).

Primero se realizará un diseño teórico usando herramientas matemáticas como Matlab, para después pasar a su implementación mediante un lenguaje propio de circuitos digitales (VHDL), no sin antes comentar los problemas encontrados al diseñar un regulador lo más rápido posible (con el menor tiempo de establecimiento posible).

3.1 Diseño teórico

Para llevar a cabo este diseño se va a partir de la función de transferencia (FDT) de un convertidor conmutado [6]:

$$G_{vd}(s) = G_{d0}\frac{(1-\frac{s}{\omega_z})}{(1+\frac{s}{Q\omega_0}+\left(\frac{s}{\omega_0}\right)^2)} \qquad (3.1.1)$$

Que tiene como entrada D (ciclo de trabajo) y como salida V (V_{out} o tensión de salida).

Para el caso de un *Buck* dichos coeficientes vienen determinados por las siguientes expresiones:

$$G_{d0} = \frac{V}{D} \qquad (3.1.2)$$

$$\omega_0 = \frac{1}{\sqrt{LC}} \qquad (3.1.3)$$

$$Q = R\sqrt{\frac{C}{L}} \qquad (3.1.4)$$

$$\omega_z = \infty \qquad (3.1.5)$$

Se puede apreciar cómo en este tipo de convertidores desaparece un cero en el numerador (por ser $\omega_z = \infty$) que sí que está presente para otras configuraciones como el *Boost* o el *Buck-Boost*.

Para la obtención de dichos coeficientes se ha tomado como tensión de entrada V_{IN} = 12 V, y como tensión de salida V_{OUT} = 5 V. Dichos valores son propios de la arquitectura *Buck*, donde $V_{IN} > V_{OUT}$.
Además, se ha escogido un valor de 22 µH para la bobina y 220 µF para el condensador.
Es decir:

$$V_{out} = V = 5V \quad (3.1.6)$$

$$V_{in} = V_g = 12V \quad (3.1.7)$$

$$D = \frac{V_{out}}{V_{in}} = \frac{5}{12} \quad (3.1.8)$$

$$L = 22\ \mu H \quad (3.1.9)$$

$$C = 220\ \mu F \quad (3.1.10)$$

Todos estos valores se corresponden con los del convertidor reductor construido por Laura Usero Puig en su TFG [7] y disponible en el laboratorio HCTLab. La topología del reductor empleado es la correspondiente a un reductor síncrono con dos interruptores controlados [Figura 3-1].

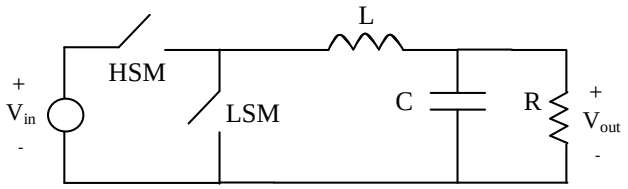


Figura 3-1: Topología del reductor

Para el caso de la R (resistencia de carga nominal), se ha procedido a realizar tres diseños:

- R = (5/8) Ω = 0,625 Ω
  Correspondiente a carga nominal para cuatro fases.
- R = (5/2) Ω = 2,5 Ω
  Correspondiente a carga nominal para una fase.
- R = 25 Ω
  Correspondiente a un 10% de carga para una fase.

Cuanto mayor es el valor de dicha resistencia (R), más inestable se vuelve el sistema. Por tanto, se puede apreciar que de los tres casos bajo estudio, el caso de R = 25 Ω es el más cercano a la inestabilidad y, por ello, el regulador final que se usará será el diseñado para este caso. Si se consigue que el sistema sea estable para el caso de mayor inestabilidad, en principio, debería ser estable para los demás casos. Aun así, se han diseñado reguladores para las demás cargas y se ha visto su comportamiento con otras cargas. Para más información sobre estos diseños se aconseja acudir a los anexos [Anexo A].

Cabe destacar que en este diseño ha primado la estabilidad por encima de la rapidez.

El último dato de interés es el periodo de muestreo que coincide con el periodo de conmutación, es decir:

$$T_{SW} = 5\ \mu s \quad (3.1.11)$$

$$f_{SW} = 200\ kHz \quad (3.1.12)$$

3.1.1 Diseño para 10% de carga para una fase

Sustituyendo dichos valores, la función de transferencia anterior (3.1.1) queda para el caso en el que R vale 25 Ω (10% de carga para una fase) de la siguiente forma:

$$G_3(s) = \frac{12}{4{,}84 \cdot 10^{-9}\, s^2 + 8{,}8 \cdot 10^{-7}\, s + 1} \qquad (3.1.13)$$

Que haciendo la conversión entre coeficientes s y z, queda:

$$G_3(z) = \frac{0{,}03097\, z + 0{,}03096}{z^2 - 1{,}994 z + 0{,}9991} \qquad (3.1.14)$$

A continuación, se ha procedido a representar su respuesta al escalón mediante el programa Matlab:

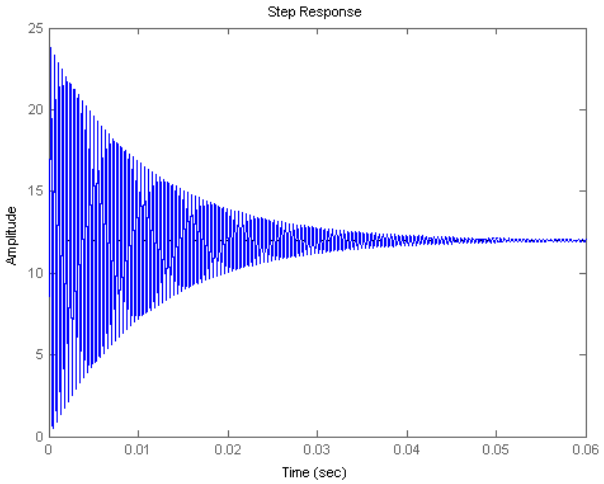


Figura 3-2: Respuesta al escalón (R = 25 Ω)

Claramente se puede apreciar como es altamente inestable.

Se ha procedido, pues, al diseño del regulador mediante la herramienta Sisotool. En este caso se ha conseguido regular el sistema añadiendo un doble cero (real) situado en 0,97 y dos polos, uno en 1 (integrador) y otro en 0. Además, se ha usado una ganancia de 0,5 que proporciona además una rapidez considerable.

El regulador diseñado queda, por tanto:

$$R_3(z) = 0{,}5\frac{(z - 0{,}97)(z - 0{,}97)}{z\,(z - 1)} \qquad (3.1.15)$$

La posición de polos y ceros anteriormente descrita se observa en la siguiente figura:

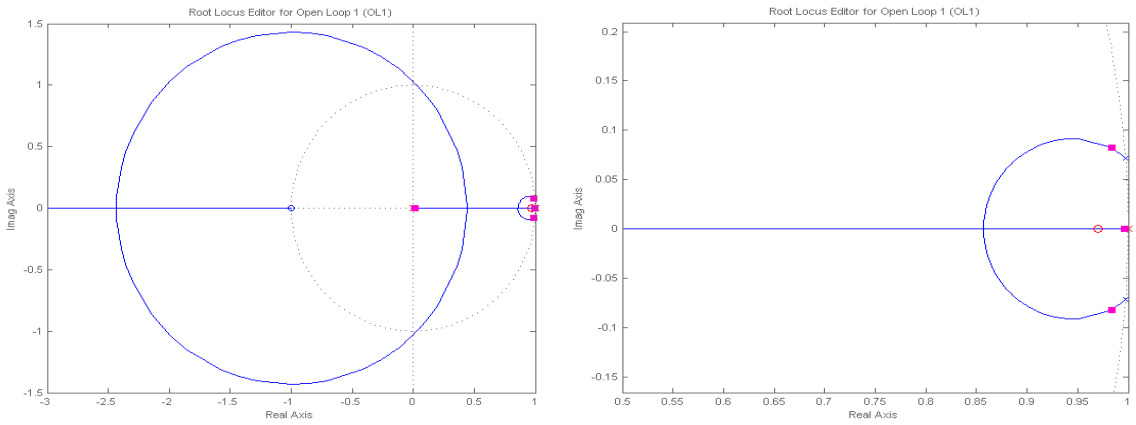


Figura 3-3: Diagrama de polos y ceros regulador R_3

Viendo la actuación (verde) y la salida en lazo cerrado (azul):

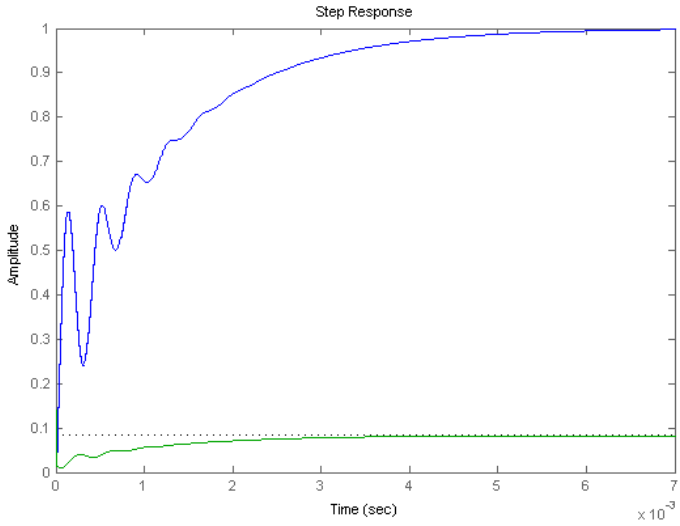


Figura 3-4: Salida en lazo cerrado y actuación regulador R_3

La dinámica es similar a un primer orden, con una pequeña oscilación al comienzo. El tiempo de establecimiento obtenido es de t_s = 3,35 ms, mayor que en los otros casos [Anexo A], pero permite que el sistema sea estable y no sobreoscile.
Si se prueba este regulador para las otras dos cargas propuestas se puede ver que es estable en ambos casos [Figura 3-5] y alcanza prácticamente los mismos tiempos de establecimiento:

- t_s = 3,3 ms para cuatro fases (R = 0,625 Ω)
- t_s = 3,34 ms para una fase (R = 2,5 Ω)

Esto se debe a que la dinámica del sistema la va a marcar el regulador de manera casi independiente a la carga que se emplee.

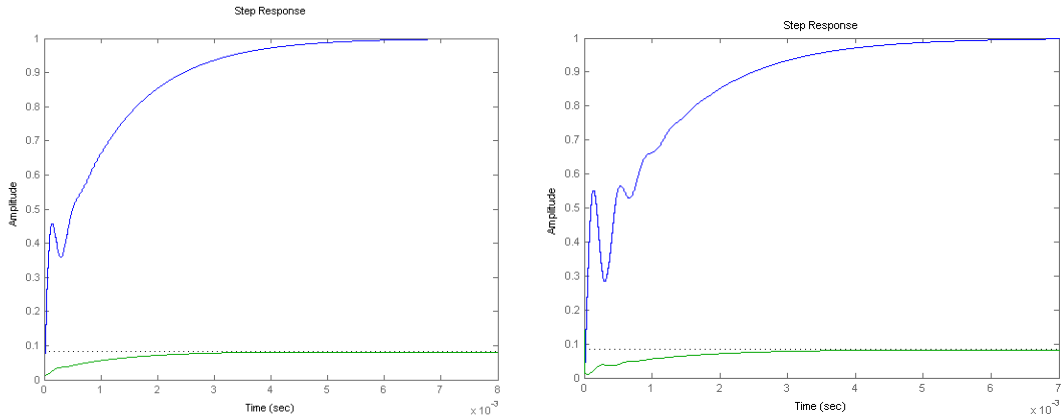


Figura 3-5: Salida en lazo cerrado y actuación (R = 0,625 Ω izda y R = 2,5 Ω dcha) con regulador R_3

A la vista de los resultados, en principio será el regulador R_3 el que se procederá a diseñar, aunque con una modificación en la ganancia debido a los efectos que se explicarán en el siguiente apartado.

3.2 Rapidez del regulador

En este apartado se va a comentar un problema detectado al implementar dicho regulador en VHDL y los cambios que se han hecho en el mismo para solventarlo.

Si directamente se usa como regulador el obtenido en el apartado anterior (3.1.15) y se incorpora al diseño VHDL realizado (que se explicará en detalle en el apartado 3.3), al simular la salida en lazo cerrado para distintos valores de consigna se obtienen los seis siguientes escalones:

- De 0 V a 5 V, de 5 V a 6 V, de 6 V a 12 V, de 12 V a 5 V, de 5 V a 0 V, de 0 V a 12 V.

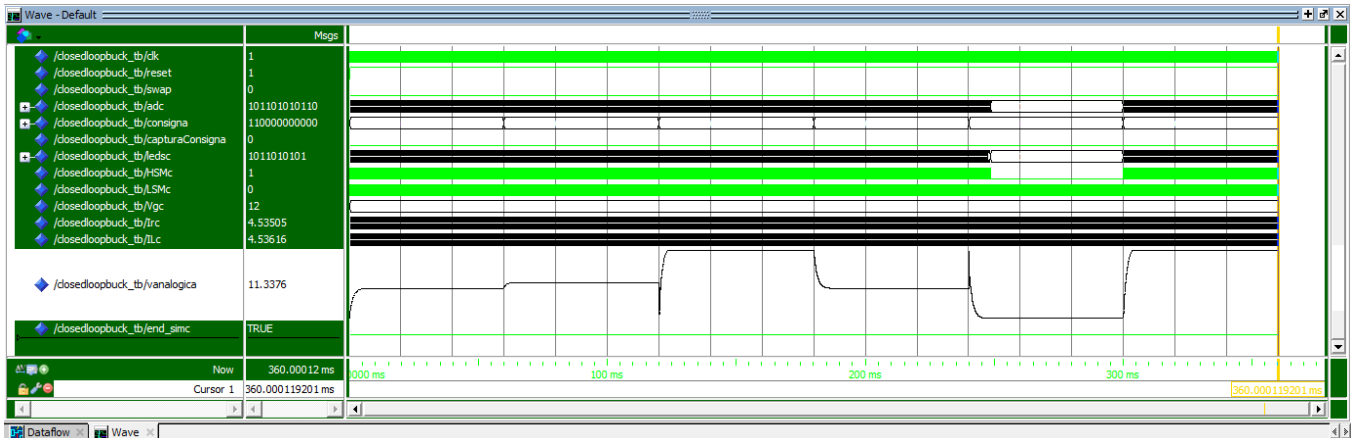


Figura 3-6: Simulación ModelSim regulador "rápido" (todos los escalones)

Se puede ver cómo para varios de los escalones aparece al comienzo un pico de subida muy pronunciado en dirección contraria a dicho escalón. Estos picos aparecen tanto en casos de subida como de bajada.

Si se toma como ejemplo el caso del escalón de 5 V a 0 V, se puede observar con mayor detalle:

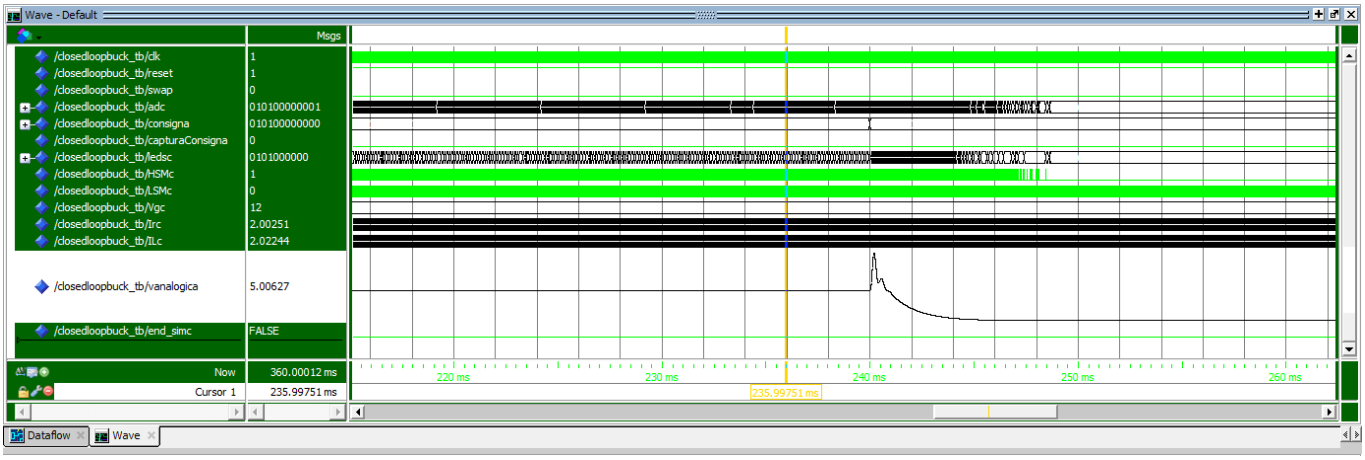


Figura 3-7: Detalle simulación ModelSim regulador "rápido" (escalón de 5 V a 0 V)

Este fenómeno se debe a que la actuación del sistema [Figura 3-4] comienza tomando en tiempo 0 un valor muy elevado, respecto al valor en régimen permanente.

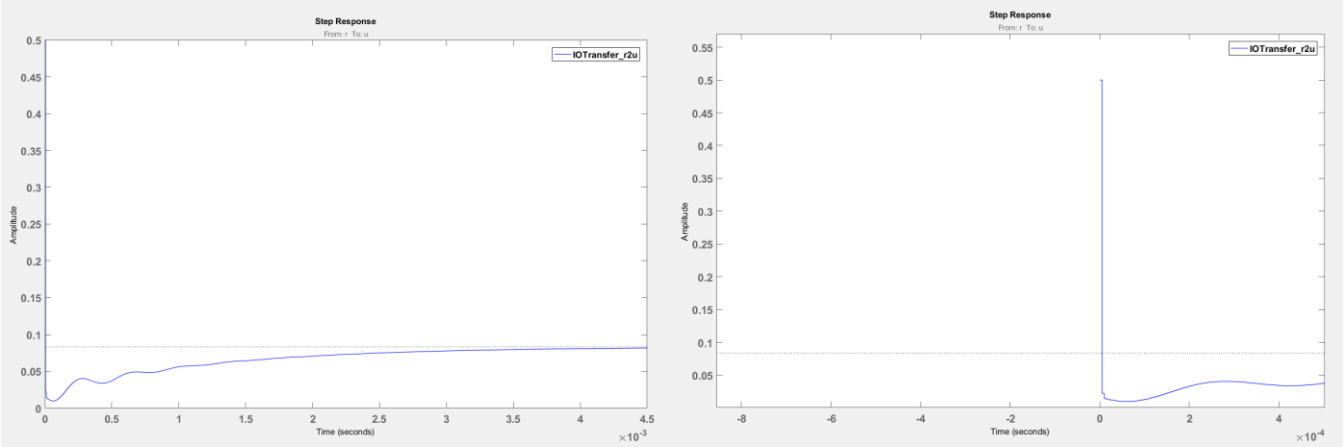


Figura 3-8: Actuación R_3

Como se puede ver [Figura 3-8], el valor inicial es 0,5, frente al valor en régimen permanente que es 0,08.
Esto da problemas ya que con un voltio el duty máximo es 0,5 (1 V => $D_{máx}$ = 0,5), mientras que con cinco voltios el duty máximo es 2,5 (5 V => $D_{máx}$ = 2,5 > 1), valor superior a 1, que es el valor de saturación (D_{sat} = 1). Debido a esto, el regulador baja desde donde no esperaba y baja más de lo esperado hasta que se da cuenta del voltaje de salida (V_{out}) y rectifica.
Por ello, para eliminar este efecto, que produce una dinámica totalmente indeseable, se ha de intentar dar con un valor de ganancia que haga que dicho valor inicial no sobrepase el valor en régimen permanente (0,08), es decir, que sea menor que el mismo.

Esto se ha logrado modificando la ganancia del regulador de 0,5 a 0,08:

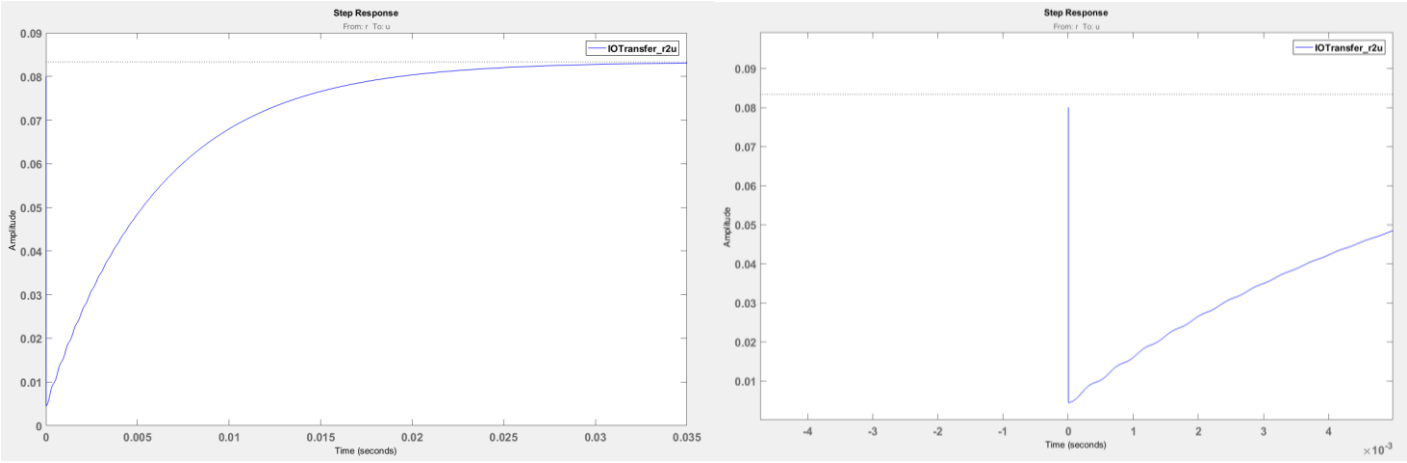


Figura 3-9: Actuación regulador final

Consiguiendo así que desaparezcan dichos picos pero sacrificando, eso sí, rapidez ya que en este caso el tiempo de establecimiento obtenido es de: t_s = 18,4 ms.

Es claramente más lento, pero como en este diseño ha primado la estabilidad frente a una gran rapidez, este es el regulador que finalmente se diseñará.

$$R_{final}(z) = 0{,}08\frac{(z-0{,}97)(z-0{,}97)}{z\,(z-1)} \qquad (3.2.1)$$

Que pasado a z^{-1} queda:

$$R_{final}(z) = \frac{0{,}08 - 0{,}1552\,z^{-1} + 0{,}075272\,z^{-2}}{1 - z^{-1}} \qquad (3.2.2)$$

La dinámica y el tiempo de establecimiento se pueden observar si representamos la salida en lazo cerrado del sistema para el caso peor (R = 25 Ω).

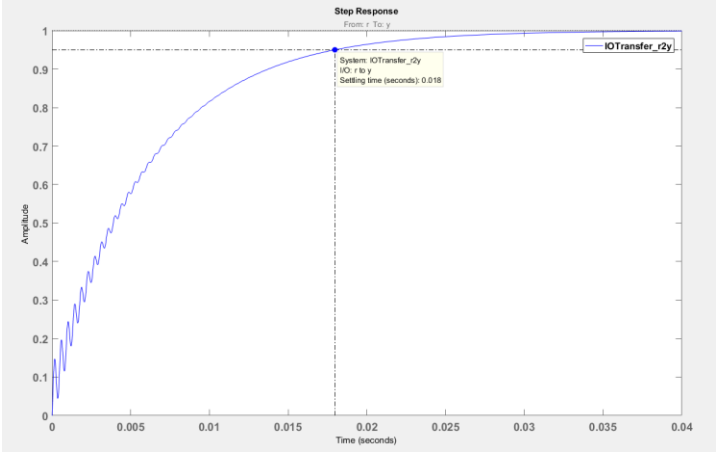


Figura 3-10: Salida en lazo cerrado regulador R_{final}

Y, por supuesto, sigue sirviendo para los otros dos casos (R = 0,625 Ω y R = 2,5 Ω), obteniendo para ambos casos prácticamente el mismo tiempo de establecimiento de 18 ms [Figura 3-11]. Esto se debe a que nuevamente el regulador es el que va a marcar la dinámica del sistema de forma casi independiente a la carga empleada.

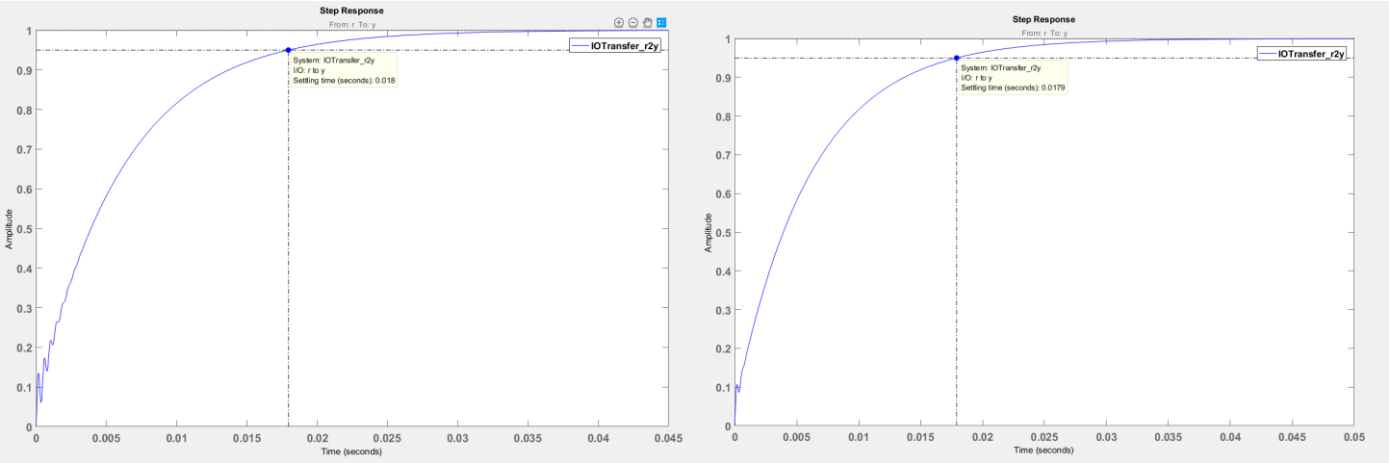


Figura 3-11: Salida en lazo cerrado (R = 0,625 Ω izda y R = 2,5 Ω dcha) con regulador R_{final}

Se puede ver cómo, aunque al principio el sistema sigue siendo algo nervioso, ha desaparecido la oscilación más pronunciada que se daba para el regulador anterior.

3.3 Implementación en VHDL

En este apartado se va a explicar cómo se ha realizado el diseño de la unidad de control incorporando el regulador anteriormente diseñado y empleando el ADC ya configurado (apartado 2.1.1), en lenguaje VHDL, mediante el programa de diseño propio de Altera: Quartus. Para ello, se empezará comentando el diseño jerárquico propuesto para el correcto funcionamiento del sistema y la coordinación de cada módulo, para pasar posteriormente a exponer la forma en que se ha generado las señales PWM y el *timing*, y se acabará comentando un estudio que se ha realizado sobre la resolución de los coeficientes del numerador del regulador.

3.3.1 Diseño jerárquico

Ahora se pasará a explicar a grandes rasgos la funcionalidad de los tres bloques fundamentales que conforman el diseño realizado:

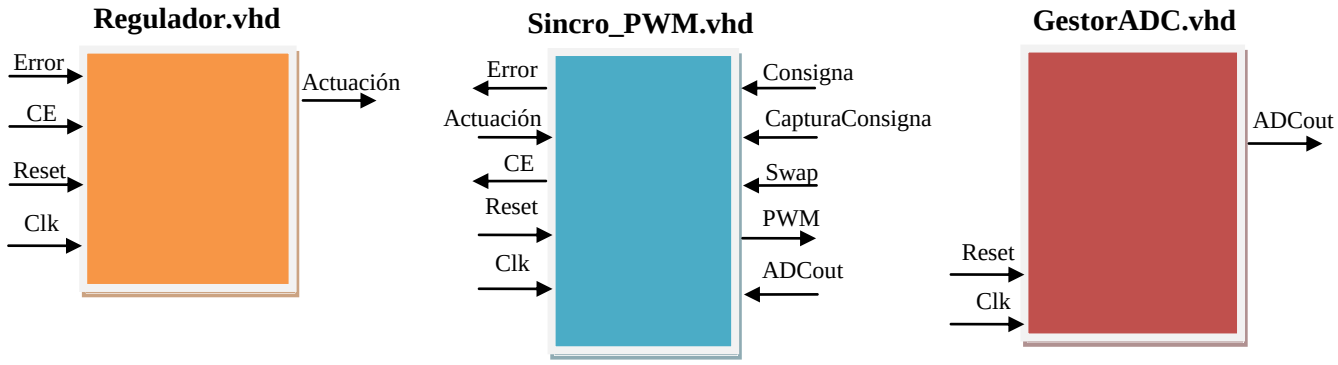


Figura 3-12: Bloques diseño VHDL

- **Regulador.vhd:**

  Este módulo contiene la implementación mediante coma fija del regulador anteriormente diseñado. Toma como entrada el error en lazo cerrado, pasa dicho error por el regulador implementado y da como salida la actuación.
  Las señales son las siguientes:

```
entity Regulador is
        port (

                Clk             : in    std_logic;
                Reset           : in    std_logic;
                CE              : in    std_logic;

                Error           : in    sfixed (4 downto -8);
                Actuacion       : out   sfixed (12 downto 0)
        );
end Regulador;
```

El criterio escogido en cuanto a la resolución de las señales de coma fija se explicará en el apartado 3.3.3.

- **Sincro_PWM.vhd:**

  Este bloque realiza varias funciones. Por un lado, se encarga de generar la señal de habilitación de los demás dispositivos o *Chip Enable* (CE). Por otro lado, genera la señal de PWM usada en cada uno de los diseños dependiendo del ciclo de trabajo que, como se verá más adelante, será HSM y LSM para una fase o HSM1, … , HSM4 y LSM1, … , LSM4 para cuatro fases.
  Además, es en Sincro_PWM.vhd donde se captura el valor de consigna si la señal de CapturaConsigna se encuentra activa.
  Por último, se encarga de calcular el error que tomará como entrada Regulador.vhd para calcular en función de este la actuación correspondiente.
  Para el cálculo del error se restan las señales de Consigna con la de ADCout, que es la salida del ADC, realizando los cambios de tipo correspondientes.
  Las señales son las siguientes (para el caso de una fase):

```
entity Sincro_PWM is
        port (

                Clk             : in    std_logic;
                Reset           : in    std_logic;

                Consigna        : in    std_logic_vector (11 downto 0);
                CapturaConsigna : in    std_logic;

                ADCout          : in    std_logic_vector (11 downto 0);
                Actuacion       : in    sfixed (12 downto 0);

                Swap            : in    std_logic;

                HSM             : out   std_logic;
                LSM             : out   std_logic;
                CE              : out   std_logic;

                error           : out   sfixed(4 downto -8)
        );
end Sincro_PWM;
```

La señal Swap se implementó como una posible forma de realizar *debugging* una vez pasado el diseño a hardware. Esta señal simplemente se encarga de poner como ciclo de trabajo la actuación que toma de la salida del Regulador.vhd o directamente la consigna para ver si la placa responde, pudiendo aplicar un ciclo de trabajo conocido
En el apartado 3.3.2 se encuentran los detalles de la generación del PWM.

- **GestorADC.vhd:**

  Bloque encargado de la gestión del ADC. Cuando detecta un dato válido, lo captura y lo pone a su salida (ADCout). Sus señales son las siguientes:

```
entity GestorADC is
        port (

                Clk             :       in      std_logic;
                Reset           :       in      std_logic;
                ADCout          :       out     std_logic_vector(11 downto 0)

        );
end GestorADC;
```

Por encima de todos ellos se encuentra como top level Controlador.vhd, cuyo cometido es la integración de los tres módulos anteriormente descritos.

```
entity Controlador is
        port (

                Clk             : in    std_logic;
                Reset           : in    std_logic;

                Swap            : in    std_logic;

                Consigna        : in    std_logic_vector (11 downto 0);

                CapturaConsigna : in    std_logic;

                ADCIn           : in    std_logic_vector (11 downto 0);

                HSM             : out   std_logic;
                LSM             : out   std_logic;

                LEDs            : out   std_logic_vector (9 downto 0)

        );
end Controlador;
```

Aquí se realizaron dos diseños: uno para simulación y otro para bajar a placa. En el de simulación se generó la señal del ADC mediante un *testbench* y, por ello, aparece en el Controlador.vhd la señal ADCIn, que es la que tomará como entrada Sincro_PWM.vhd. Mientras que en el caso de bajar a placa, esta señal desaparece ya que el propio ADC toma como entrada analógica la señal presente en el pin de entrada correspondiente y, por ello, no es necesaria señal de ADCIn y bastará con conectar la salida del Gestor_ADC.vhd con la entrada de Sincro_PWM.vhd.
En cuanto a la señal LEDs, se encargará de mostrar en los LEDs de la placa el valor del ADC.
Con lo que el diagrama del diseño jerárquico propuesto quedaría de la siguiente forma:

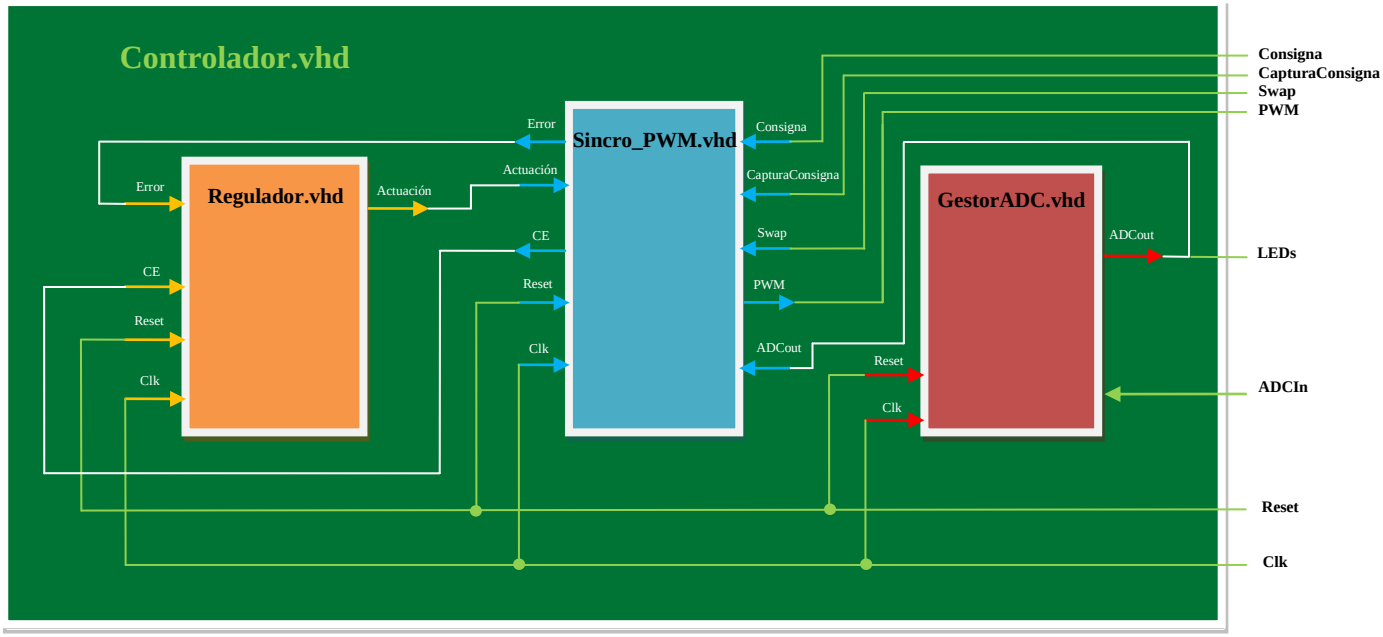


Figura 3-13: Diagrama diseño VHDL

3.3.2 *Timing* y generación de PWM

A continuación, se pasará a explicar la forma en que se ha planeado el *timing* del sistema.

Para empezar, en Sincro_PWM.vhd se ha creado una señal de 8 bits que actúa como contador de 0 a 255. Esto se debe a que se ha empleado un reloj de 50 MHz con un periodo de conmutación de 200 kHz, por lo que el ciclo de trabajo será de 50 MHz / 200 kHz = 250 que usando la potencia de dos más cercana 2^8 queda 256 (N = 256).

```
signal contador              :  unsigned (7 downto 0) := "00000000";
```

Otro proceso se encarga de comprobar el valor de dicha señal de modo que cuando el contador llegue a 251 se activa al ciclo siguiente la señal de *Chip Enable* (CE). Es decir, CE se encuentra activo en el ciclo N-4 o, lo que es lo mismo, en el ciclo 252.

```
process (Clk,Reset)
begin
   if Reset = '0' then
      CE <= '0';
   elsif rising_edge (Clk) then
      if contador = "11111011" then
         CE <= '1';
      else
         CE <= '0';
      end if;
   end if;
end process;
```

Por otro lado, en Regulador.vhd se da a cada señal empleada el tamaño deseado y se inicializan los coeficientes del numerador del regulador propuesto (B0, B1 y B2) en el instante N-5 (ciclo 251). De este modo, en N-4 (ciclo 252) que, como se ha visto, es el

momento en el que CE se encuentra activo, se realiza la realimentación de los valores anteriores a las nuevas entradas a emplear.

```
elsif (rising_edge(clk)) then
   if (CE = '1') then
      sX0 <= sX;
      sX1 <= sX0;
      sX2 <= sX1;

      sY1 <= sY_6_27Reg;
      sY2 <= sY1;

   end if;
```

Y, además, en este ciclo es en el que realiza la multiplicación de los coeficientes y se obtiene la entrada, que es el error, y la actuación:

```
-- Entrada
sX <= Error;

-- Salida
Actuacionc <= to_slv(sY);
Actuacion <= to_sfixed(Actuacionc,12,0);


-- Cálculo de las operaciones intermedias
sB0X0 <= resize(sX0 * B0, sB0X0);
sB1X1 <= resize(sX1 * B1, sB1X1);
sB2X2 <= resize(sX2 * B2, sB2X2);

sA1Y1 <= resize(sY1 * A1, sA1Y1);
```

En el siguiente ciclo (N-3 ó 253) se han registrado dichos valores:

```
elsif (rising_edge(clk)) then

   sB0X0Reg <= sB0X0;
   sB1X1Reg <= sB1X1;
   sB2X2Reg <= sB2X2;

   sA1Y1Reg <= sA1Y1;

end if;
```

A continuación, se ha calculado el valor de la salida actual y se ha limitado la salida para que tome un valor entre 0 y 0,95:

```
-- Calcular el valor de la salida actual

sY_6_27 <= resize (sB0X0Reg + sB1X1Reg + sB2X2Reg - sA1Y1Reg, sY_6_27);


--Limitamos sY_6_27

sY_6_27int <=  (others => '0')       when sY_6_27 < limite_inferior else
               limite_superior       when sY_6_27 > limite_superior else
               sY_6_27;
```

En N-2 (ciclo 254) se ha registrado la salida para su futura realimentación:

```
elsif (rising_edge(clk)) then

   sY_6_27Reg <= sY_6_27int;

end if;
```

Por último, se ha transformado dicho valor al formato de la salida:

```
-- Transformar el valor al tamaño de salida

sY <= resize (sY_6_27Reg,sY);
```

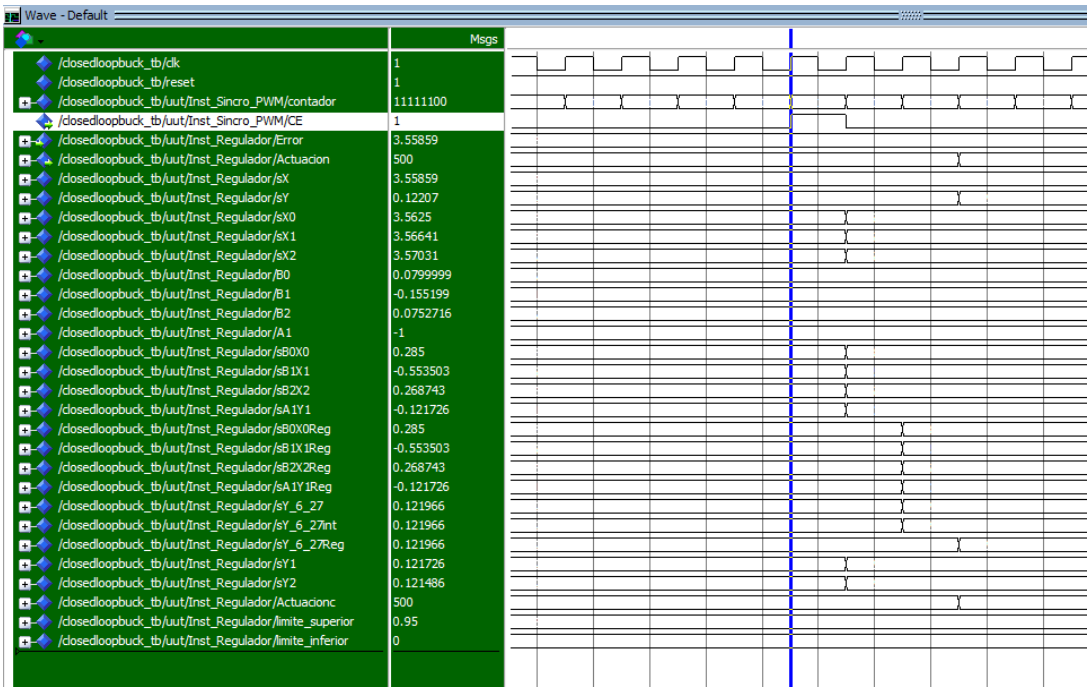


Figura 3-14: Ejemplo *timing* simulación ModelSim

En la figura anterior [Figura 3-14] se pueden apreciar con un ejemplo todas las transiciones de *timing* anteriormente comentadas desde el ciclo 252 en que se activa CE en adelante.

En cuanto a la generación del PWM para el caso de una fase se han usado dos señales correspondientes a los dos interruptores del circuito: HSM y LSM [Figura 3-1]. Estas señales son señales alternadas, de modo que cuando una se encuentra a 1, la otra a 0 y viceversa. Además, se ha tenido que dejar un pequeño diferencial de tiempo de dos ciclos entre la subida de una de ellas y la bajada de la otra, ya que hay un breve instante en el que ambas valen 1 y esto no es admisible, ya que ambos interruptores no pueden estar activos a la vez. Esta técnica se conoce como tiempo muerto (*Dead Time*) y consiste en un breve tiempo con ambas señales de control a 0 para evitar conducción cruzada en los dos interruptores.

```
process (Clk,Reset)
begin
   if Reset = '0' then
      HSM <= '0';
      LSM <= '0';
   elsif rising_edge (Clk) then
      if contador < unsigned (CicloTrabajo) then
         HSM <= '1';
      else
         HSM <= '0';
      end if;
      if (contador > unsigned (CicloMasDT)) and (contador < unsigned(MaxContDT)) then
         LSM <= '1';
      else
         LSM <= '0';
      end if;
   end if;
end process;
```

En el caso de la versión para cuatro fases, cada fase tiene sus dos interruptores HSM y LSM, por lo que tendremos en total ocho señales.
La idea es que cada una de las fases se encuentre desfasada un cuarto de periodo respecto a la anterior. Esto se puede conseguir simplemente sumando a cada contador el valor del contador anterior más un cuarto del valor máximo del contador (256/4 = 64) [Figura 3-15].

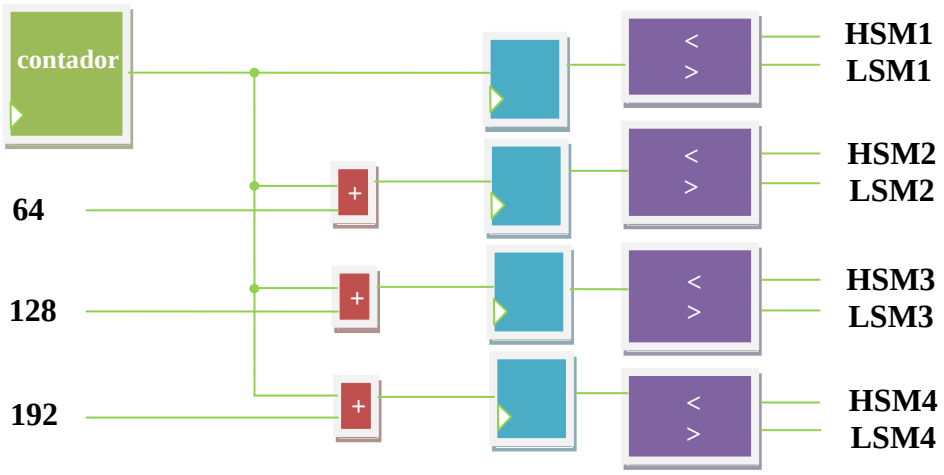


Figura 3-15: Esquema generación señales HSM y LSM alternadas

Con los resultados obtenidos en simulación se puede comprobar que se ha conseguido que cada par HSM y LSM nunca se encuentren activos al mismo tiempo [Figura 3-16].

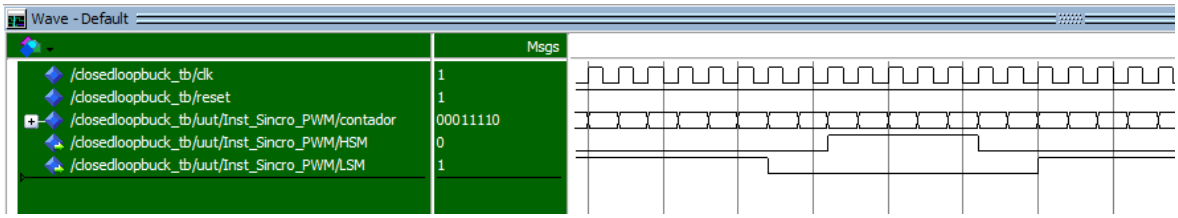


Figura 3-16: Ejemplo diferencial de tiempo entre HSM y LSM

Así mismo, se ha logrado generar cada HSM o LSM con un desfase de un cuarto de periodo (64 ciclos) respecto a su fase anterior. Como se puede comprobar en la siguiente figura [Figura 3-17], el tiempo que transcurre entre la subida de una señal HSM de una fase y la subida de la señal HSM de la fase siguiente es siempre de 1,28 µs, correspondiente a los 64 ciclos de diferencia entre ellas. Además se puede observar en la diferencia de 64 unidades siempre presente entre los contadores de las distintas fases.

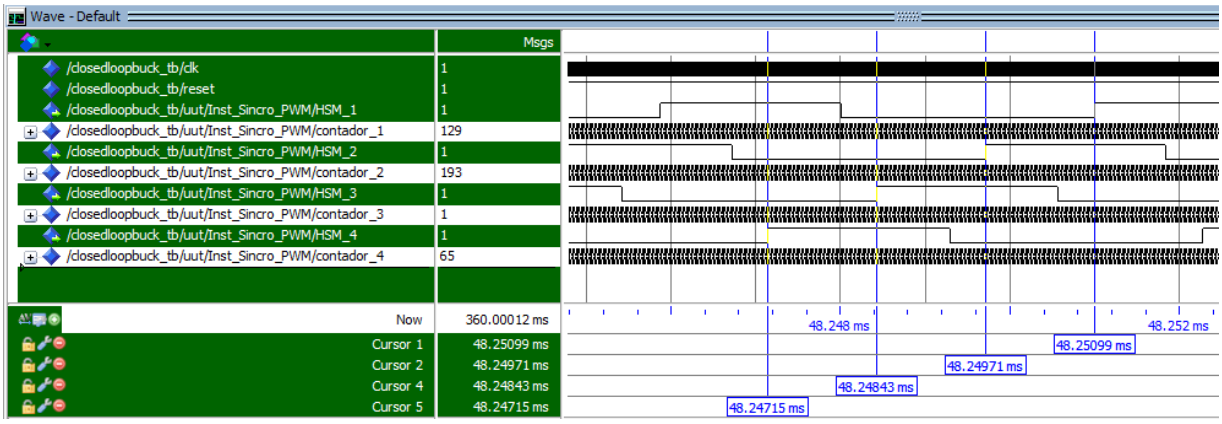


Figura 3-17: Ejemplo de desfase entre señales HSM (cuatro fases)

En la figura siguiente [Figura 3-18] se puede ver el funcionamiento habitual de las señales HSM y LSM en una simulación:

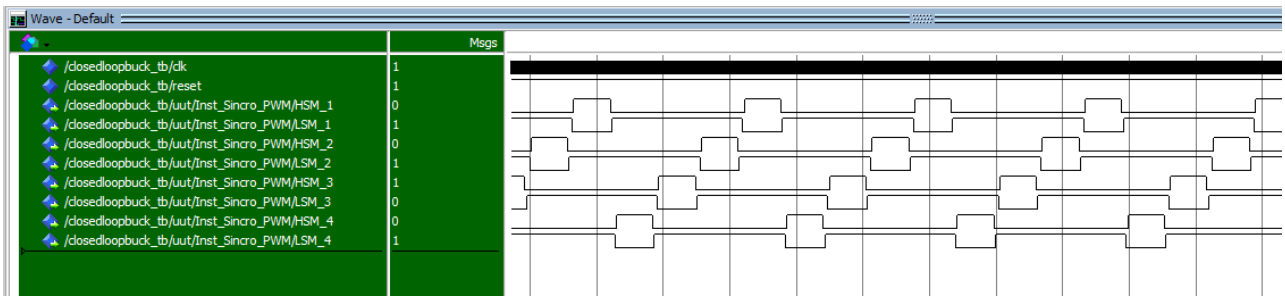


Figura 3-18: Ejemplo *timing* PWM (cuatro fases)

Para más información acerca de la implementación del código VHDL se recomienda recurrir al Manual del Programador [Anexo B].

3.3.3 Resolución coeficientes regulador

Si se parte del regulador diseñado y se trata de traducir sus coeficientes a un lenguaje como VHDL para su integración digital, uno de los principales retos a los que se enfrenta el diseñador viene a la hora de elegir la resolución de cada uno de sus coeficientes.
En este trabajo se ha estudiado el efecto que produce en la dinámica y en el rendimiento del sistema el hecho de escoger una resolución u otra para los coeficientes del numerador del regulador y se ha llegado a la conclusión de que dicha elección es crítica para conseguir el funcionamiento deseado.

En el caso de un convertidor *Buck*, sus dos polos se encuentran cerca del círculo unidad debido a que su frecuencia de conmutación es superior a la del filtro RLC asociado. Como consecuencia de esto, los ceros del regulador se suelen encontrar también cerca de los polos de la planta y del círculo unidad. Por ello, no solo se debe tener en cuenta la posición de los ceros del regulador, sino su posición relativa respecto a otros ceros, a los polos de la planta y al círculo unidad. Esta posición relativa es determinante para el rendimiento del regulador, ya que un mínimo cambio en la posición de los ceros supone una modificación en la respuesta del regulador y en la dinámica del sistema.

Un controlador PID (Controlador Proporcional-Integral-Derivativo), como es el caso del diseñado (3.2.2), tiene una función de transferencia del tipo:

$$R(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - z^{-1}} \qquad (3.3.1)$$

Que pasado a su forma de ecuación en diferencias queda:

$$y_k = y_{k-1} + b_0 x_k + b_1 x_{k-1} + b_2 x_{k-2} \qquad (3.3.2)$$

En el método conocido como “Direct Form” [8] o forma directa se emplean directamente dichos coeficientes para la implementación hardware mediante coma flotante o coma fija (en el caso de este TFG se emplea coma fija).

Hay varios métodos para comprobar la resolución correcta de los coeficientes. Mientras que algunas fuentes proponen soluciones más triviales [9] como evitar que la suma de los coeficientes del numerador sea cero ($\sum b_i \neq 0$) y que la suma de los del denominador sea cero ($\sum a_i = 0$), otras fuentes sugieren métodos más exhaustivos [10].
Una forma sencilla que se ha empleado en este TFG para caracterizar la resolución de los coeficientes del numerador ha sido mediante el cálculo de su sumatorio, una vez que estos han sido cuantificados con la resolución escogida, que no es más que la salida del regulador ante entrada unidad:

$$\sum \tilde{b}_i \qquad (3.3.3)$$

Esta suma debe ser diferente de cero como ya se ha comentado. Lo que además se puede hacer con este sumatorio es compararlo con el correspondiente al caso ideal, es decir, con los coeficientes teóricos del regulador diseñado y de este modo ver el error de cuantificación:

$$\sum b_i \qquad (3.3.4)$$

Esta comparación ofrece una forma simple pero eficaz de comprobar a partir de qué momento los resultados teóricos y prácticos convergen y, por lo tanto, indica el punto cuasi óptimo en el cual no es necesario aumentar más la resolución y, con ello, no incrementar los recursos requeridos.
En el módulo Regulador.vhd se han declarado e inicializado dichos coeficientes: B0, B1 y B2 (en el ejemplo con una resolución de 20 bits):

```
--coeficientes
signal B0 : sfixed (0 downto -19) := to_sfixed(0.08, 0 , -19);
signal B1 : sfixed (0 downto -19) := to_sfixed(-0.1552, 0, -19);
signal B2 : sfixed (0 downto -19) := to_sfixed(0.075272, 0, -19);
```

Se va a comprobar qué sucede si se emplean distintas resoluciones para ocho casos de estudio que se corresponden con resoluciones de 8, 12, 14, 16, 18, 20, 22 y 24 bits.
Para cada una de dichas resoluciones se va a mostrar a continuación una simulación del escalón de 0 V a 5 V, así como el valor cuantificado de los coeficientes del numerador y, en los casos que corresponda, el tiempo de establecimiento al 5%, es decir a 4,75 V.

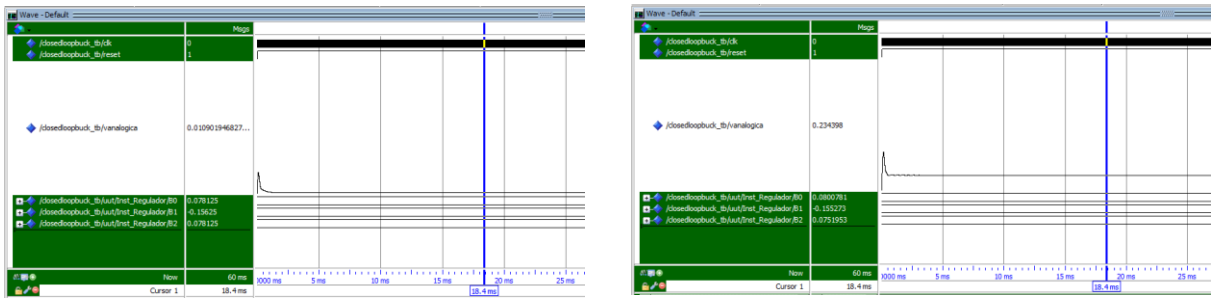


Figura 3-19: Escalón de 0 V a 5 V y coeficientes B0, B1 y B2: 8 bits (izda) y 12 bits (dcha)

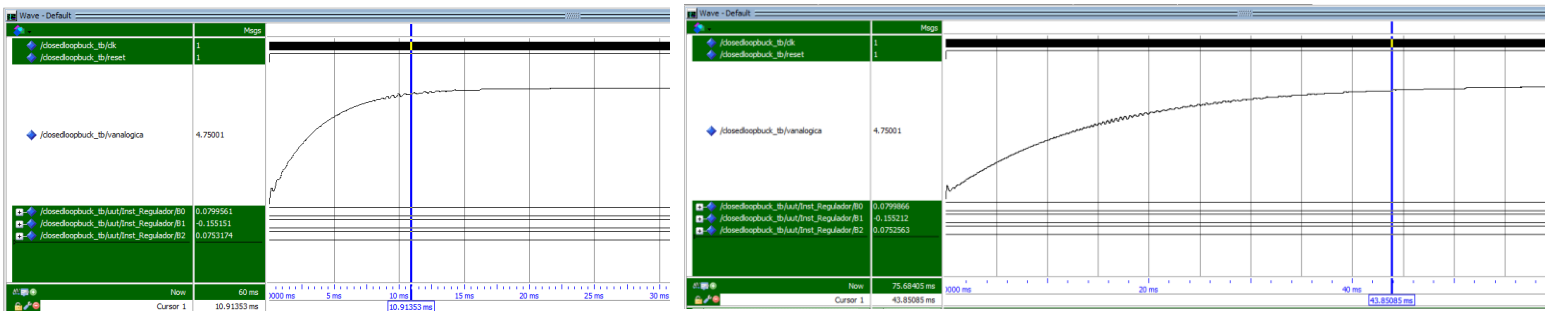


Figura 3-20: Escalón de 0 V a 5 V, t_s y coeficientes B0, B1 y B2: 14 bits (izda) y 16 bits (dcha)

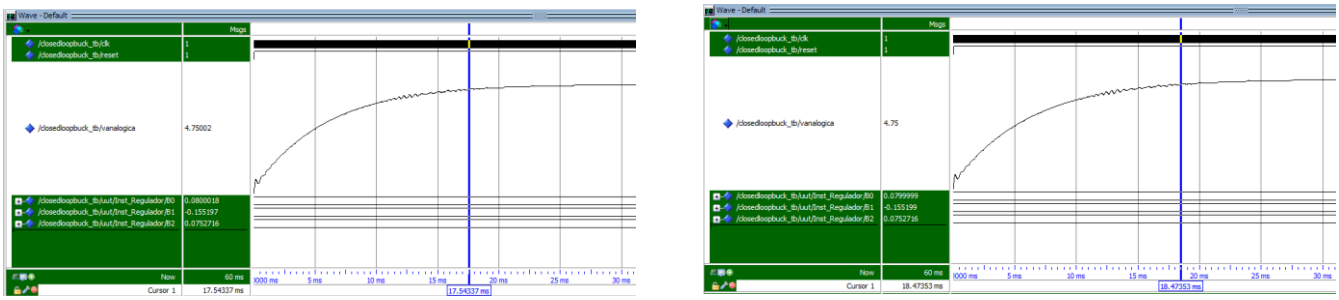


Figura 3-21: Escalón de 0 V a 5 V, t_s y coeficientes B0, B1 y B2: 18 bits (izda) y 20 bits (dcha)

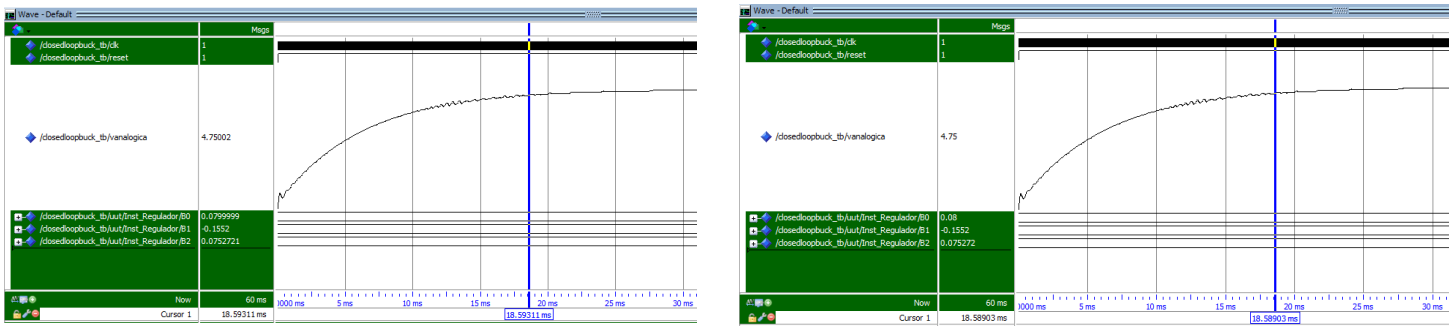


Figura 3-22: Escalón de 0 V a 5 V, t_s y coeficientes B0, B1 y B2: 22 bits (izda) y 24 bits (dcha)

Nótese que todos estos resultados que se presentan se corresponden al caso de una única fase con R = 2,5 Ω.

A fin de tener todos los resultados obtenidos de forma organizada se ha elaborado la siguiente tabla, en la que se puede observar el valor de cada coeficiente (B0, B1 y B2), la suma de los mismos y el tiempo de establecimiento tanto para cada una de las resoluciones $R_8(z)…R_{24}(z)$ como para el caso ideal $R_i(z)$:

	$R_i(z)$	$R_8(z)$	$R_{12}(z)$	$R_{14}(z)$	$R_{16}(z)$	$R_{18}(z)$	$R_{20}(z)$	$R_{22}(z)$	$R_{24}(z)$
B_0	0,08	0,0800781	0,0800781	0,0799561	0,0799866	0,0800018	0,0799999	0,0799999	0,08
B_1	-0,1552	-0,154297	-0,155273	-0,155151	-0,155212	-0,155197	-0,155199	-0,1552	-0,1552
B_2	0,075272	0,0742188	0,0751953	0,0753174	0,0752563	0.0752716	0,0752716	0,0752721	0,075272
$\sum_i B_i$	0,000072	-0,0000001	-0,0032419	0,0001225	0,0000309	0,0000764	0,0000725	0,000072	0,000072
$t_s(ms)$	18,4	---	---	10,91353	43,85085	17,54337	18,47353	18,59311	18,58903

Tabla 1: Coeficientes numerador, sumatorio y t_s para todas las resoluciones

A la vista de los resultados ya es mucho más evidente por qué es necesario dar con la resolución adecuada en cada diseño.

Para los dos primeros casos (8 bits y 12 bits) no tiene sentido hablar de tiempo de establecimiento [Figura 3-19] ya que directamente no se obtiene el escalón esperado. Esto se debe a que, en el caso de 8 bits, sus coeficientes suman prácticamente 0, mientras que en el caso de 12 bits la suma toma un valor negativo. Por tanto, no tiene sentido diseñar con estas resoluciones.

Para el caso siguiente de 14 bits se puede ver que la suma de los coeficientes es bastante superior a la del caso ideal, por ello, el tiempo de establecimiento es de casi 11 ms, valor inferior a los 18,4 ms teóricos. Es por tanto más rápido en este caso.

Con 16 bits sucede justo lo contrario; la suma es casi la mitad que la ideal y eso hace que el tiempo de establecimiento sea de casi 44 ms, algo más del doble que en el caso ideal.

Es a partir de 18 bits cuando el comportamiento del sistema parece acercarse al esperado, con un sumatorio de 0,0000764, bastante próximo a 0,000072, que hace que el tiempo de establecimiento sea ya de casi 18 ms.

Ya con 20 bits los sumatorios y los tiempos empiezan a converger, hasta que subimos a 24 bits, momento en el cual los coeficientes y la suma son iguales a los del caso ideal. Se puede ver, sin embargo, que el valor de simulación al que parece que converge el tiempo de establecimiento es de unos 18,59 ms, que es algo diferente a los 18,4 ms obtenidos teóricamente. Esta diferencia se debe principalmente al modelado del *Buck* implementado para la simulación y la función de transferencia teórica empleada. Es un error más que asumible por temas de modelado.

Se ha visto, pues, la enorme importancia que tiene escoger la resolución correcta de coma fija en un diseño de control digital. Es totalmente decisivo para conseguir el comportamiento deseado. En el siguiente apartado se seguirá incidiendo en este tema, ya que se mostrarán los resultados obtenidos en simulación y en montaje hardware para cada una de las resoluciones anteriormente estudiadas.

4 Integración, pruebas y resultados

En este apartado se va a exponer la forma en la que se ha montado el circuito para realizar las medidas pertinentes. Además, se mostrarán los resultados obtenidos de manera experimental y se compararán con los conseguidos mediante simulación, tanto para el caso de una fase como para el de cuatro fases y para todas las resoluciones.

4.1 Montaje

Para el montaje del circuito se han usado los siguientes componentes:

- Placa Terasic DE10-Lite con FPGA MAX10 10M50DAF484C7G (apartado 2.1).

Figura 4-1: Placa Terasic DE10-Lite con FPGA MAX10

- Convertidor *Buck* de cuatro fases diseñado por Laura Usero Puig en su TFG [7] y disponible en el HCTLab de la EPS, con los valores ya comentados (apartado 3.1): tensión de entrada V_{IN} = 12 V, tensión de salida V_{OUT} = 5 V, corriente de salida I_{OUT} = 8 A, con una potencia total máxima por tanto de 40 W y con 2 A por cada fase (10 W por fase).

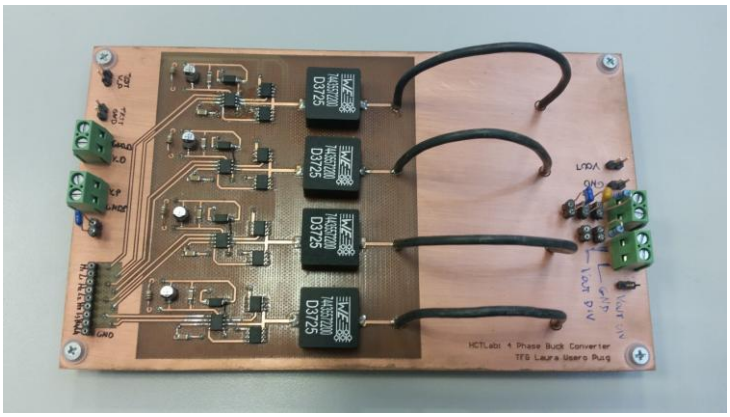


Figura 4-2: Convertidor *Buck* de cuatro fases

- Cables para la realización de las conexiones, tanto pares trenzados para conseguir pocas interferencias como cables individuales gruesos para las conexiones por las que circula una mayor corriente y, por tanto, con mayor disipación de potencia.

- Carga variable mediante interruptores [7] con un rango de valores de entre aproximadamente 0,474 Ω y 100 Ω.

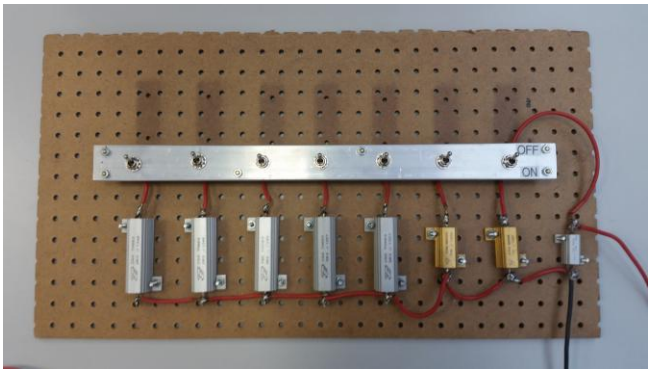


Figura 4-3: Carga variable mediante interruptores

- Osciloscopio de señal mixta Agilent MSO-X 3014A con sondas tanto de corriente como de tensión, así como sondas digitales.

Figura 4-4: Osciloscopio de señal mixta Agilent MSO-X 3014A

- Fuente de alimentación TTI EX354Tv Triple Power Supply 300W con tensión máxima de 35 V y corriente máxima de 4 A.

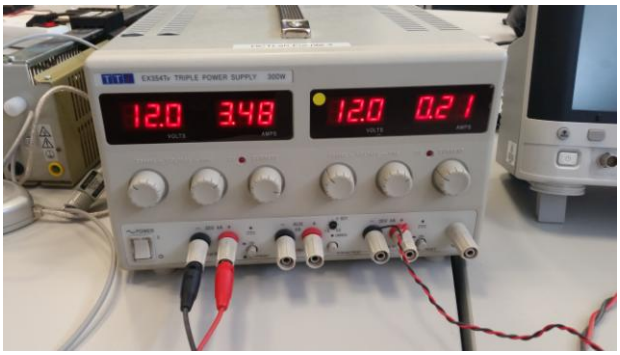


Figura 4-5: Fuente de alimentación TTI EX354Tv Triple Power Supply 300W

Las conexiones entre ellos se realizaron de la siguiente forma:

- Las señales HSM y LSM del reductor se conectaron a pines digitales I/O de la placa [Figura 4-6]. En el caso de una fase, las señales HSM/LSM no utilizadas se conectaron a tierra mediante una *protoboard*. El GND del reductor fue conectado a su vez al GND de la placa para tener la misma referencia de masa. Para el reloj o Clk del sistema se escogió el pin correspondiente al reloj de 50 MHz. Se conectaron los nueve bits más significativos de la Consigna a los switches de la placa y el Swap al switch restante. Reset y CapturaConsigna se conectaron a los dos botones de los que dispone la placa [11].

Node Name	Direction	Location
Clk	Input	PIN_P11
Consigna[11]	Input	PIN_F15
Consigna[10]	Input	PIN_B14
Consigna[9]	Input	PIN_A14
Consigna[8]	Input	PIN_A13
Consigna[7]	Input	PIN_B12
Consigna[6]	Input	PIN_A12
Consigna[5]	Input	PIN_C12
Consigna[4]	Input	PIN_D12
Consigna[3]	Input	PIN_C11
Consigna[2]	Input	
Consigna[1]	Input	
Consigna[0]	Input	
HSM_1	Output	PIN_AB7
HSM_2	Output	PIN_AB9
HSM_3	Output	PIN_AA11
HSM_4	Output	PIN_AB17
LEDs[9]	Output	PIN_B11
LEDs[8]	Output	PIN_A11
LEDs[7]	Output	PIN_D14
LEDs[6]	Output	PIN_E14
LEDs[5]	Output	PIN_C13
LEDs[4]	Output	PIN_D13
LEDs[3]	Output	PIN_B10
LEDs[2]	Output	PIN_A10
LEDs[1]	Output	PIN_A9
LEDs[0]	Output	PIN_A8
LSM_1	Output	PIN_AB8
LSM_2	Output	PIN_Y10
LSM_3	Output	PIN_AA12
LSM_4	Output	PIN_AA17
Reset	Input	PIN_B8
Swap	Input	PIN_C10

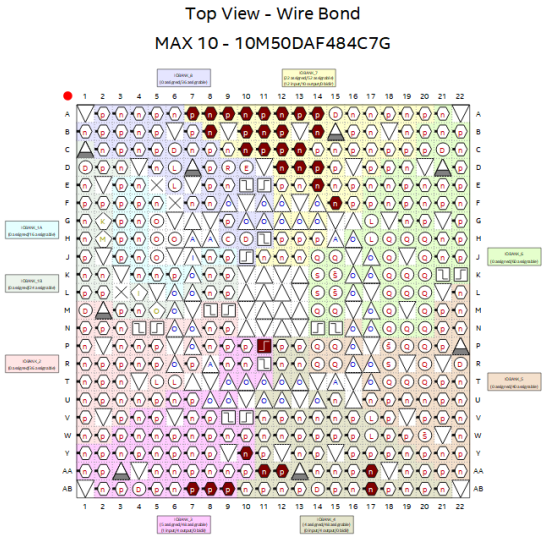


Figura 4-6: Asignación de pines cuatro fases

- La alimentación del circuito de potencia (V_P y GND_P) se conectó a una de las fuentes a 12 V mediante cables gruesos con una limitación de corriente de algo menos de 3 A.
- La alimentación del driver (V_ D y GND_D) se conectó a otra de las fuentes a 12 V mediante un par trenzado, dejando pasar una corriente mucho menor.
- La salida (VOUT) se conectó mediante cables gruesos a la carga variable.
- La salida tras la división (VOUT DIV) se conectó mediante un par trenzado al pin analógico de entrada del ADC de la FPGA: ADC_IN0 y la masa a un pin de GND cercano (nuevamente, conectado al pin de tierra del *Buck*).
- Las sondas del osciloscopio: la de medida de tensión en el VOUT del reductor y en su GND correspondiente, y la de medida de corriente conectada al cable de salida de la fase a medir.
- FPGA conectada a un ordenador para ser programada con Quartus.

El montaje queda mucho más claro ilustrado mediante el esquema elaborado [Figura 4-7].

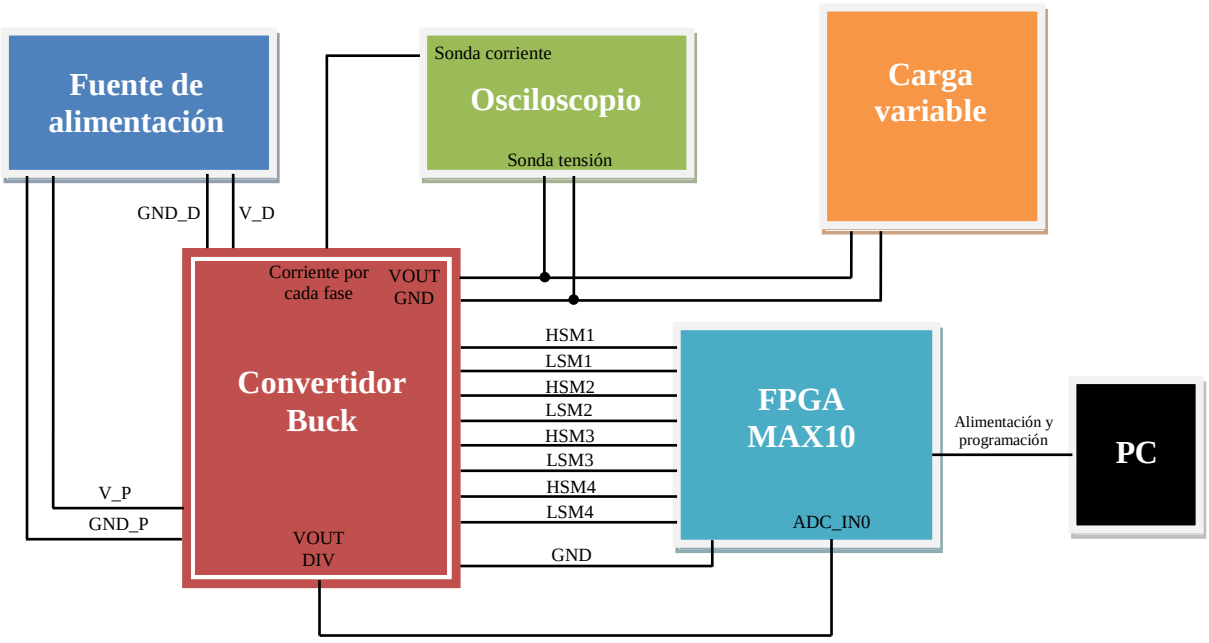


Figura 4-7: Esquema montaje cuatro fases

Además, se incluye una foto de una medida típica realizada en el laboratorio de un escalón entre 0 V y 5 V con cuatro fases y una resistencia de carga de 0,625 Ω [Figura 4-8].

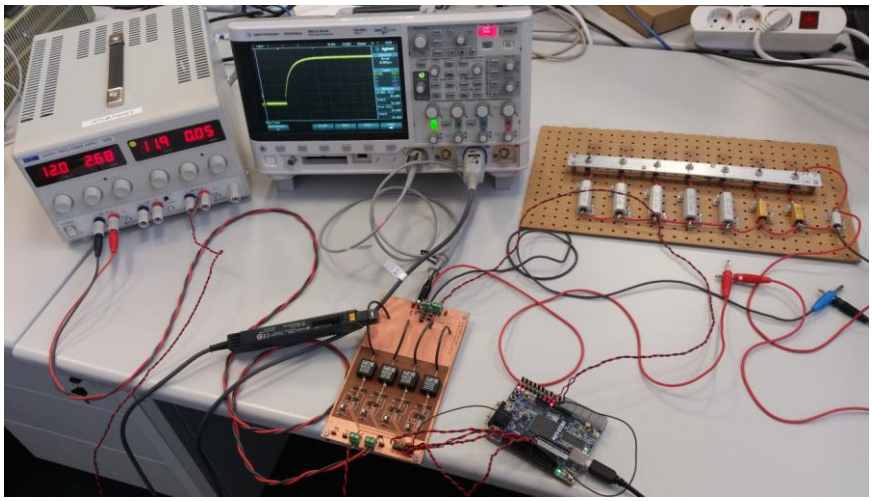


Figura 4-8: Ejemplo de montaje para cuatro fases en HCTLab

4.2 Regulador Buck para una fase

En este apartado se verán las peculiaridades y los resultados obtenidos para el caso de una única fase, tanto en simulación como en hardware.

4.2.1 Resultados simulación

Para el caso de una fase se ha realizado la simulación empleando como planta el modelo del reductor presente en el archivo BuckMonofaseReal.vhd diseñado por los profesores Ángel de Castro y Alberto Sánchez [7]. Este módulo cuenta con las siguientes señales:

```
entity BuckMonofaseReal is
    port (
        --In
        Clk : in std_logic;
        Reset : in std_logic;
        HSM : in std_logic; -- On = '1', off = '0'
        LSM : in std_logic; -- On = '1', off = '0'
        Vg : in real;
        Ir : in real;
        -- Out
        IL : out real;
        Vout : out real
    );
end BuckMonofaseReal;
```

Como se puede ver este módulo toma del *testbench* elaborado (ClosedLoopBUCK_tb.vhd) [Anexo B] las señales de Clk, Reset, HSM y LSM, conectadas a las señales correspondientes de Controlador.vhd. A Vg se le ha dado un valor de 12 V mientras que Ir se calcula como la división entre el voltaje analógico de entrada al ADC entre el valor de la resistencia de carga que, en el caso de una fase, es de 2,5 Ω. En cuanto a las salidas a IL, no se le ha dado uso, mientras que Vout es el nuevo valor de entrada que va tomando el ADC. Este *testbench* además hace de ADC traduciendo el valor analógico de voltaje de entrada a un valor digital empleado como valor de ADC por Controlador.vhd.

Los resultados de simulación para una fase ya se han mostrado en el apartado 3.3.3, donde se ha visto el tiempo de establecimiento, la dinámica y la suma de coeficientes para cada una de las ocho resoluciones bajo estudio [Tabla 1] de los escalones de entre 0 V y 5 V, obtenidos teniendo en cuenta las consideraciones anteriores. Por ello, para el caso unifase se pasará directamente a mostrar los resultados experimentales.

4.2.2 Resultados experimentales

En el caso de una fase las conexiones se han realizado tal cual se han mostrado con anterioridad [Figura 4-7][Figura 4-8] con la particularidad de que las señales HSM2, HSM3, HSM4, LSM2, LSM3 y LSM4 se han conectado a GND dejando únicamente una fase activa. Tras dar un valor de carga de 2,5 Ω y programar la FPGA se procedió a visualizar en el osciloscopio los escalones de entre 0 V y 5 V obtenidos a la salida al operar sobre la consigna (manejada por los switches).

A la vista de los resultados de la simulación [Tabla 1] se ha creído conveniente realizar el montaje para las resoluciones de 14 bits y superiores, ya que las resoluciones de 8 y 12 bits no dan escalón alguno.

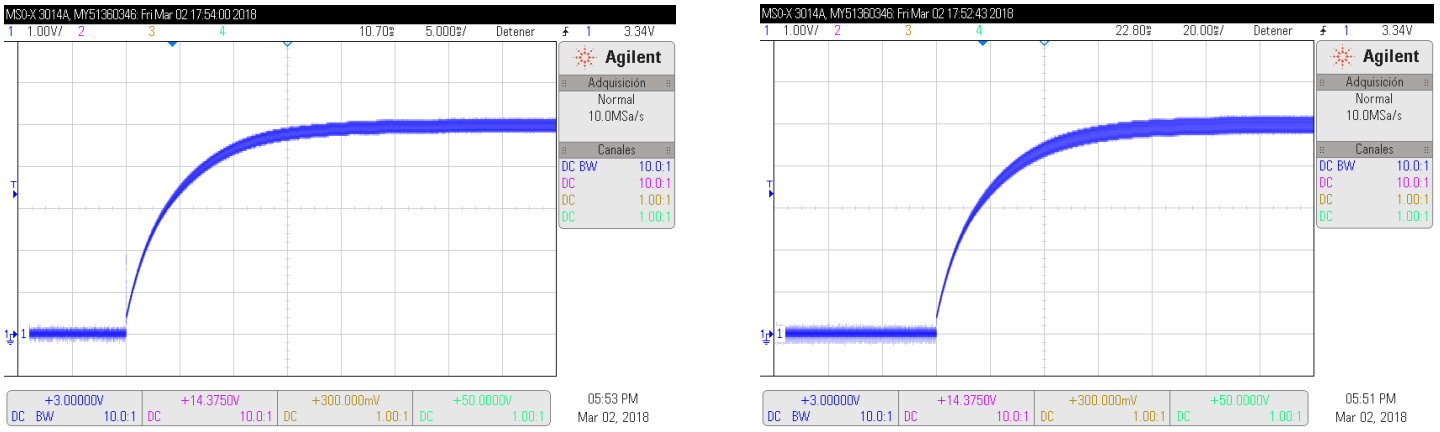


Figura 4-9: Escalón 0 V a 5 V una fase resolución: 14 bits (izda) y 16 bits (dcha)

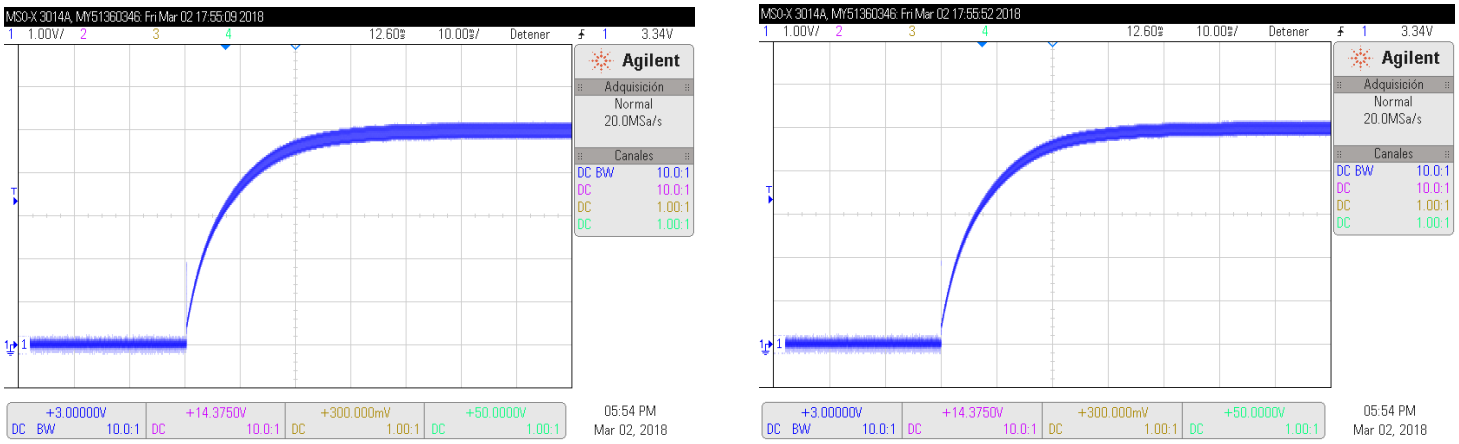


Figura 4-10: Escalón 0 V a 5 V una fase resolución: 18 bits (izda) y 20 bits (dcha)

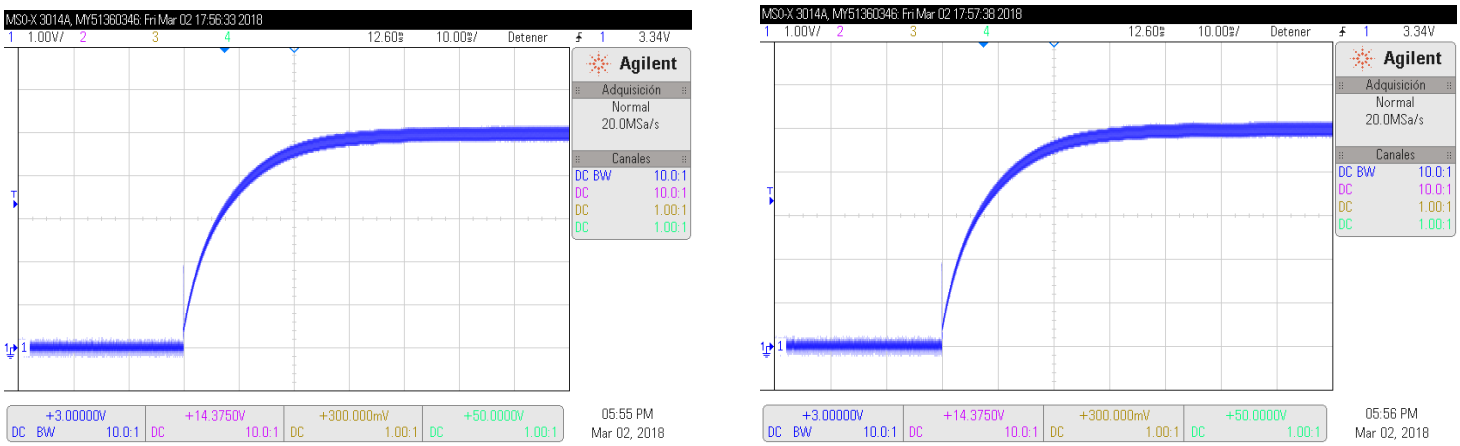


Figura 4-11: Escalón 0 V a 5 V una fase resolución: 22 bits (izda) y 24 bits (dcha)

Se puede comprobar que estas figuras dan unos resultados muy similares a los resultados de simulación [Figuras 3-20 a 3-22] presentes en la tabla realizada (Tabla 1). Se puede ver cómo para el caso de 14 bits (5 ms por división) el establecimiento al 5% (4,75 V) da de manera experimental un valor de algo más de 12 ms, cuando en simulación se obtenía un valor de unos 11 ms, mientras que para el caso de 16 bits (20 ms por división), que en simulación daba un t_s de casi 44 ms, en hardware se consigue un valor algo mayor.

Para el resto de resoluciones (ya todas con 10ms por división), que son las que daban valores más cercanos a los del regulador diseñado, mientras que en simulación se obtenía en casi todos los casos tiempos de establecimiento de unos 18 ms, experimentalmente tiene un valor de unos 20 ms, incluso superior en algunos casos.
Estas pequeñas diferencias se deben a temas de modelado, ya que a la hora de simular se ha empleado un reductor "virtual" frente a un reductor real en el montaje, además con componentes no ideales. El circuito se encuentra también expuesto tanto al ruido como a otros factores externos que degradan ligeramente la calidad de la señal.
Sin embargo, los resultados experimentales se asemejan mucho a los teóricos, con los errores típicos de una implementación hardware real. Por tanto, se puede afirmar que se trata de un montaje satisfactorio.

4.3 Regulador Buck para cuatro fases

A continuación, se mostrarán y se discutirán los resultados obtenidos para el caso de cuatro fases tanto en simulación como al bajar el diseño a hardware.

4.3.1 Resultados simulación

Para el caso de cuatro fases se ha realizado la simulación empleando como planta un modelo del reductor similar al empleado en el caso anterior, pero adaptado a la versión de cuatro fases (BuckMultifaseReal.vhd), disponible en el Manual del Programador [Anexo B]. Este módulo cuenta con las siguientes señales:

```
entity BuckMultifaseReal is
   port (
      --In
      Clk : in std_logic;
      Reset : in std_logic;
      HSM_1 : in std_logic;
      LSM_1 : in std_logic;
      HSM_2 : in std_logic;
      LSM_2 : in std_logic;
      HSM_3 : in std_logic;
      LSM_3 : in std_logic;
      HSM_4 : in std_logic;
      LSM_4 : in std_logic;
      Vin   : in real;
      Ir    : in real;
      --Out
      Iin   : out real;
      Vout  : out real
   );
end BuckMultifaseReal;
```

Que nuevamente toma del *testbench* las señales de Clk y Reset, así como las señales HSM y LSM de cada fase, conectadas a las señales correspondientes de Controlador.vhd [Figura 3-15]. A Vin se le ha dado un valor de 12 V, mientras que Ir se calcula como la división del voltaje analógico de entrada al ADC entre el valor de la resistencia de carga que, en el caso de cuatro fases, es de 0,625 Ω.
En cuanto a las salidas, a la corriente Iin no se le ha dado uso, mientras que Vout es el nuevo valor de entrada que va tomando el ADC.
Esta nueva versión del *testbench* sigue haciendo también la función del ADC traduciendo el valor analógico de voltaje de entrada a un valor digital empleado como valor de ADC por Controlador.vhd.

Si se vuelven a simular escalones de 0 V a 5 V para distintas resoluciones los resultados obtenidos son muy similares a los vistos en el apartado 3.3.3 para el caso de una fase, con ligeras diferencias en los tiempos de establecimiento, pero con una dinámica prácticamente igual, ya que viene marcada por el regulador.

Mientras que, para los casos de baja resolución, donde se siguen consiguiendo resultados no deseados [Figura 4-12], como para el caso de 8 bits, donde ni siquiera se lograba un escalón o el caso de 14 bits, donde se sigue obteniendo un t_s menor al esperado (unos 10 ms), para el caso de resoluciones aceptables se llega a unos resultados bastante buenos, como en el caso de 20 bits [Figura 4-13], donde se consigue un t_s de unos 18,4 ms, muy cercano al deseado.

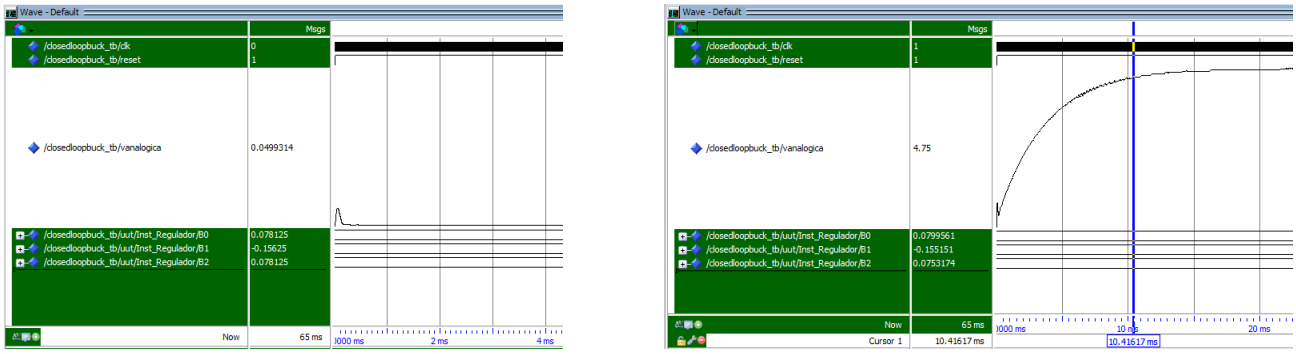


Figura 4-12: Escalón de 0 V a 5 V, t_s y coeficientes B0, B1 y B2: 8 bits (izda) y 14 bits (dcha)

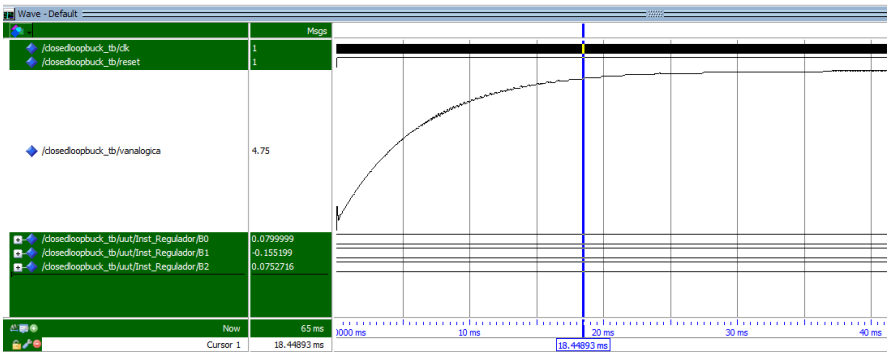


Figura 4-13: Escalón de 0 V a 5 V, t_s y coeficientes B0, B1 y B2: 20 bits

Mucho más interesante es observar el desfase que aparece entre las corrientes de cada una de las fases. Dado que se desea a la salida un valor de corriente lo más lineal posible, es decir, con el menor rizado posible, las corrientes que hay en cada una de las fases deben estar desfasadas para que al sumarlas los rizados se compensen. El desfase para cuatro fases es de 90º por cada fase, luego las fases tendrán desfases de 0º, 90º, 180º y 270º. Esto se puede apreciar en las siguientes figuras [Figura 4-14][Figura 4-15].

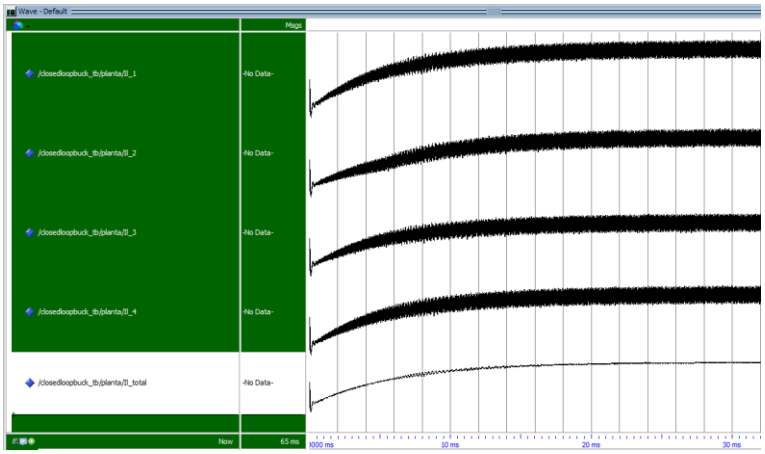


Figura 4-14: Corrientes por cada fase y corriente total

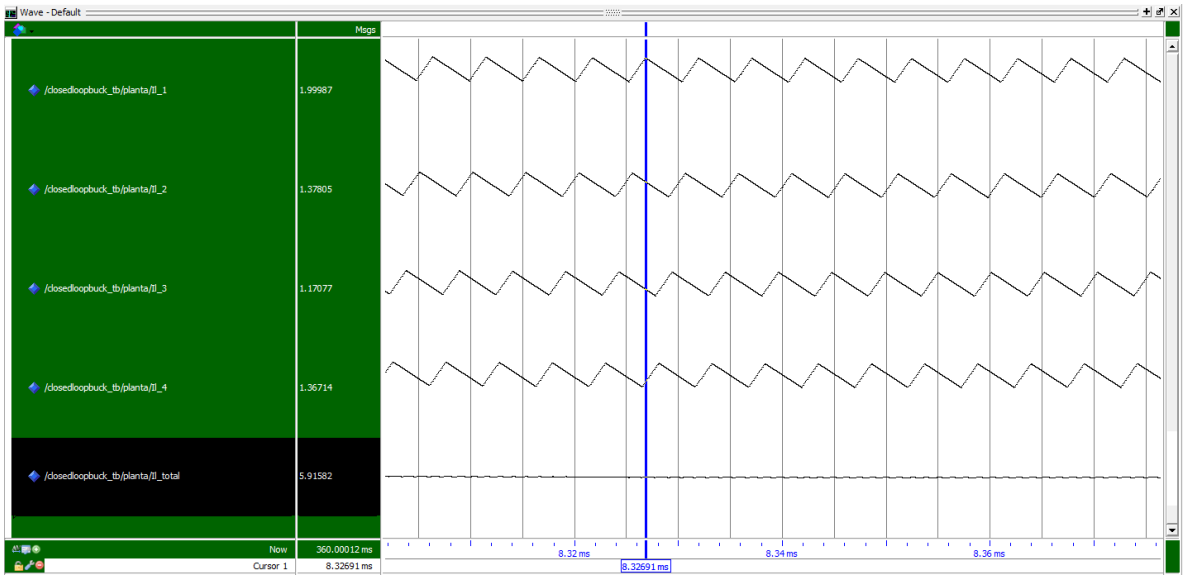


Figura 4-15: Detalle corrientes por cada fase y corriente total

4.3.2 Resultados experimentales

Tras realizar el montaje descrito para cuatro fases [Figura 4-7][Figura 4-8] y programar la FPGA, se procedió a visualizar en el osciloscopio los escalones de entre 0 V y 5 V obtenidos a la salida al operar sobre la consigna (manejada por los switches).
Se muestran las medidas realizadas para dos resoluciones "bajas" con las que se obtenían valores no deseados: 14 bits [Figura 4-16] y 16 bits [Figura 4-17], así como los resultados para la resolución a la que los resultados teóricos y prácticos empezaban a converger razonablemente: 20 bits [Figura 4-18]. Los resultados aparecen para las dos cargas utilizadas. Acompañado de cada escalón se representa la corriente medida por una de las fases en cada uno de los casos.

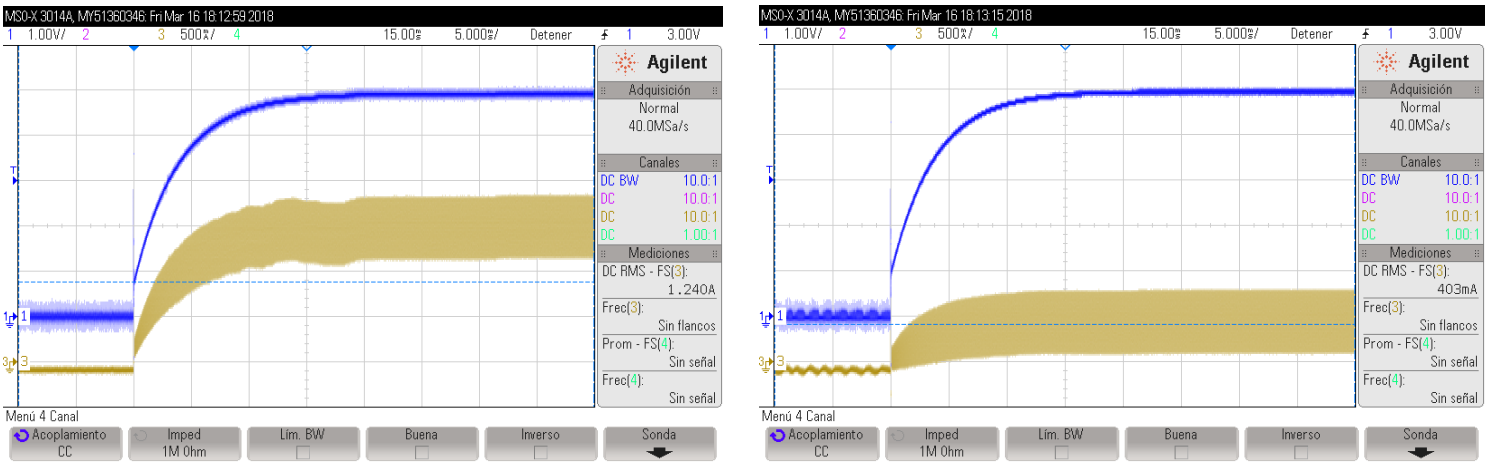


Figura 4-16: Escalón 0 V a 5 V cuatro fases resolución 14 bits carga: 0,625 Ω (izda) y 2,5 Ω (dcha) y corriente por una fase

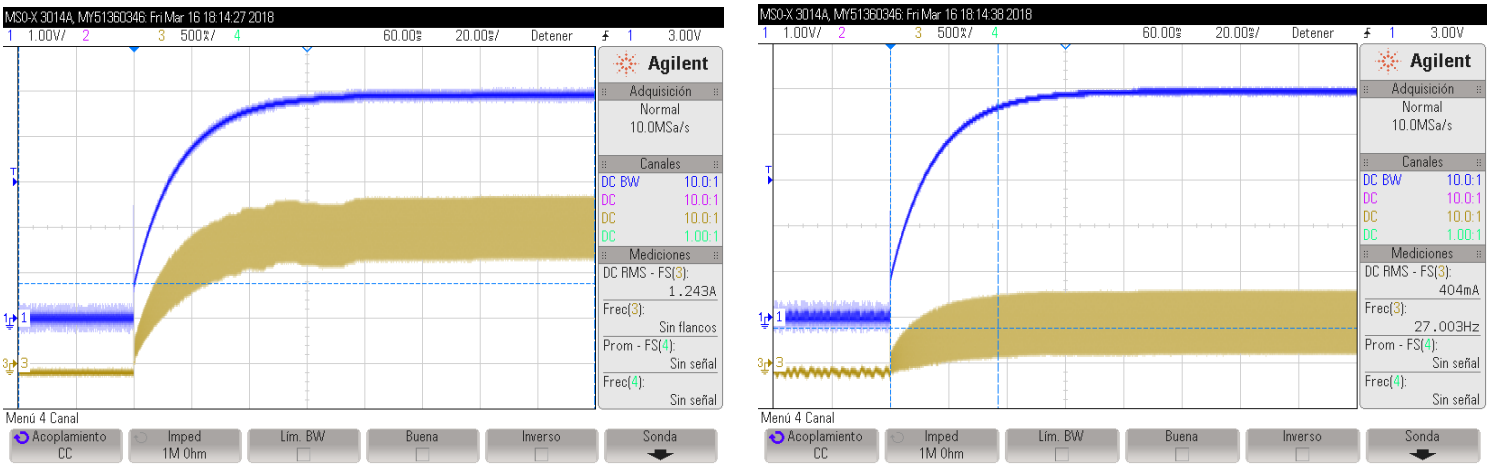


Figura 4-17: Escalón 0 V a 5 V cuatro fases resolución 16 bits carga: 0,625 Ω (izda) y 2,5 Ω (dcha) y corriente por una fase

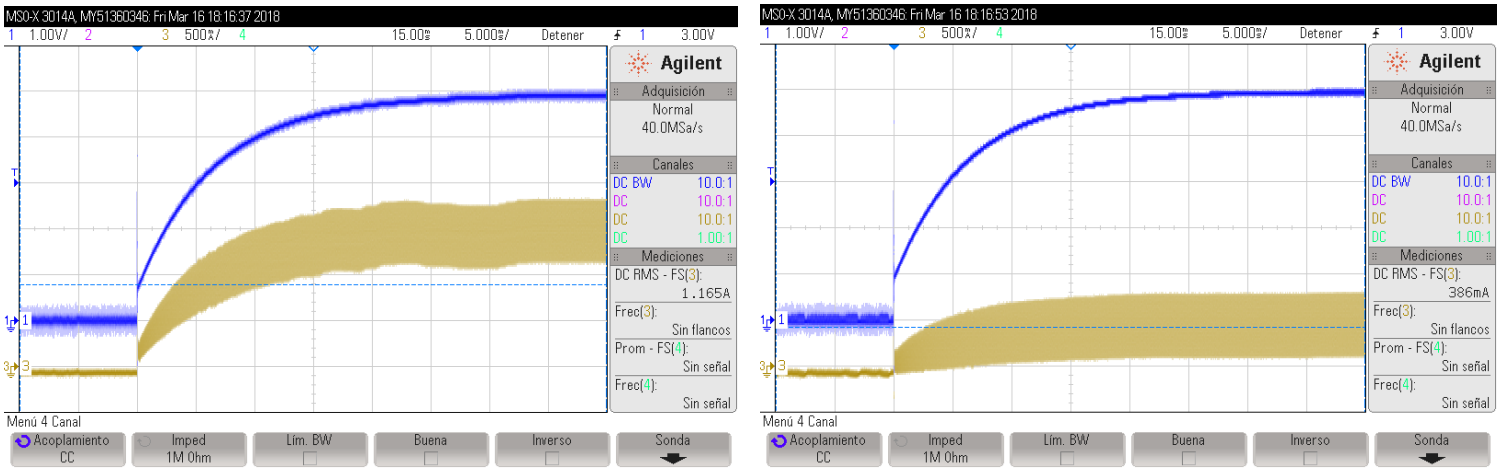


Figura 4-18: Escalón 0 V a 5 V cuatro fases resolución 20 bits carga: 0,625 Ω (izda) y 2,5 Ω (dcha) y corriente por una fase

Los resultados obtenidos en hardware se puede observar que son muy similares a los conseguidos en simulación (apartado 4.3.1). Por un lado, cabe destacar que en los casos de baja resolución se siguen teniendo tiempos de establecimiento diferentes al del regulador diseñado. En el caso de 14 bits (5 ms por división), de alrededor de 10 ms y para 16 bits (20 ms por división), de unos 40 ms, mientras que para el caso de 20 bits se consigue un t_s cercano al deseado, que es de unos 18 ms. Nuevamente aparecen ligeros errores en estos tiempos, que son en todos los casos algo mayores a los esperados, debido a los temas de modelado en hardware anteriormente comentados (apartado 4.2.2). Sin embargo, se aproximan lo suficiente a los teóricos para poder afirmar que el montaje ha sido exitoso.

Por otro lado, en estas figuras se puede observar además la influencia de la carga que se conecta al circuito. Si bien es cierto que apenas se modifica el escalón obtenido al variar la carga, ya que la dinámica la marca el regulador y es prácticamente igual en ambos casos, las corrientes por cada fase son mucho mayores cuanto más pequeña es la resistencia de carga, alcanzando valores cercanos a 1,2 A en el caso de R = 0,625 Ω frente a los 0,4 A para R = 2,5 Ω. Esto se debe a que al disminuir la resistencia y dejar constante la tensión, la corriente debe aumentar para compensar esa nueva resistencia de un valor menor.

A continuación, se va a proceder a analizar las corrientes por cada fase.

Debido a que únicamente se disponía de dos sondas para medir corrientes, las medidas se han realizado observando el desfase entre fases de dos en dos.
Si se mide la corriente en dos de las fases se puede apreciar que, pese a tener valores muy parecidos, hay ligeras diferencias entre una fase y otra [Figura 4-19].

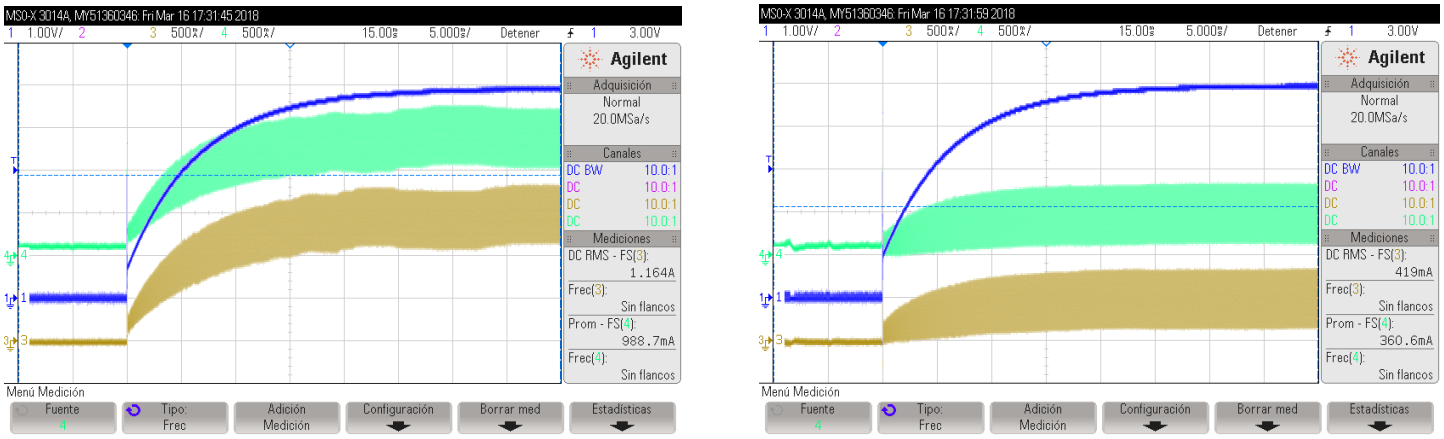


Figura 4-19: Escalón 0 V a 5 V cuatro fases resolución 20 bits carga: 0,625 Ω (izda) y 2,5 Ω (dcha) y corriente por dos de las cuatro fases

Por otro lado, si se toma la primera de las fases como referencia y se va midiendo la corriente en las demás [Figura 4-20][Figura 4-21], se puede ver de manera experimental el desfase de 90º entre corrientes ya comentado (apartado 4.3.1), necesario para la compensación de los rizados y, con ello, para la obtención de una corriente casi lineal a la salida.

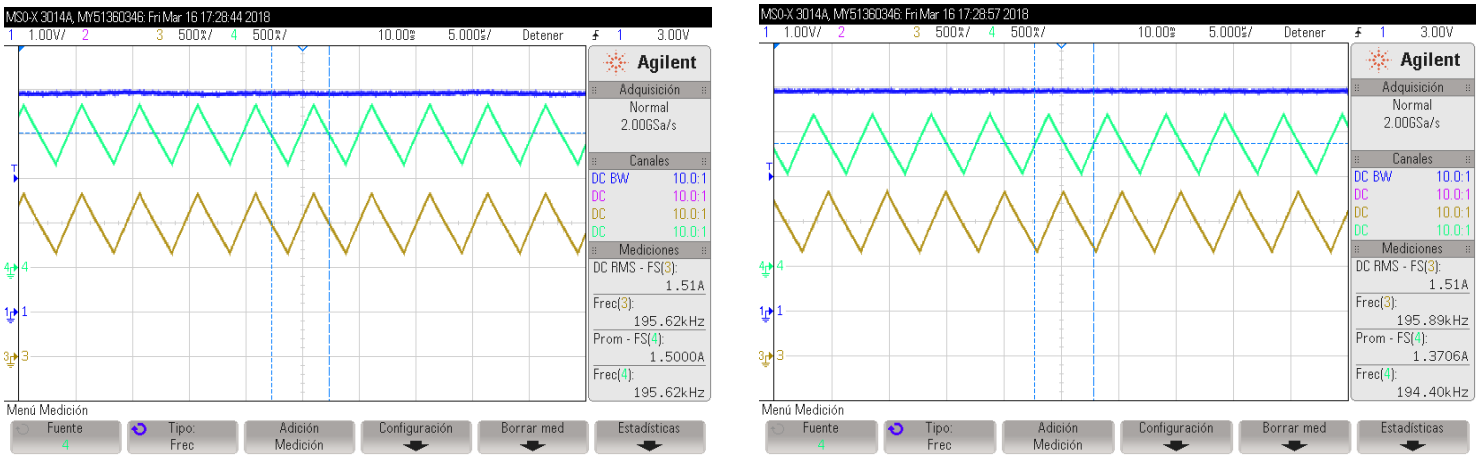


Figura 4-20: Desfase entre corrientes: fase 1/fase 1 (izda) y fase 1/fase 2 (dcha), resolución 20 bits carga 0,625 Ω

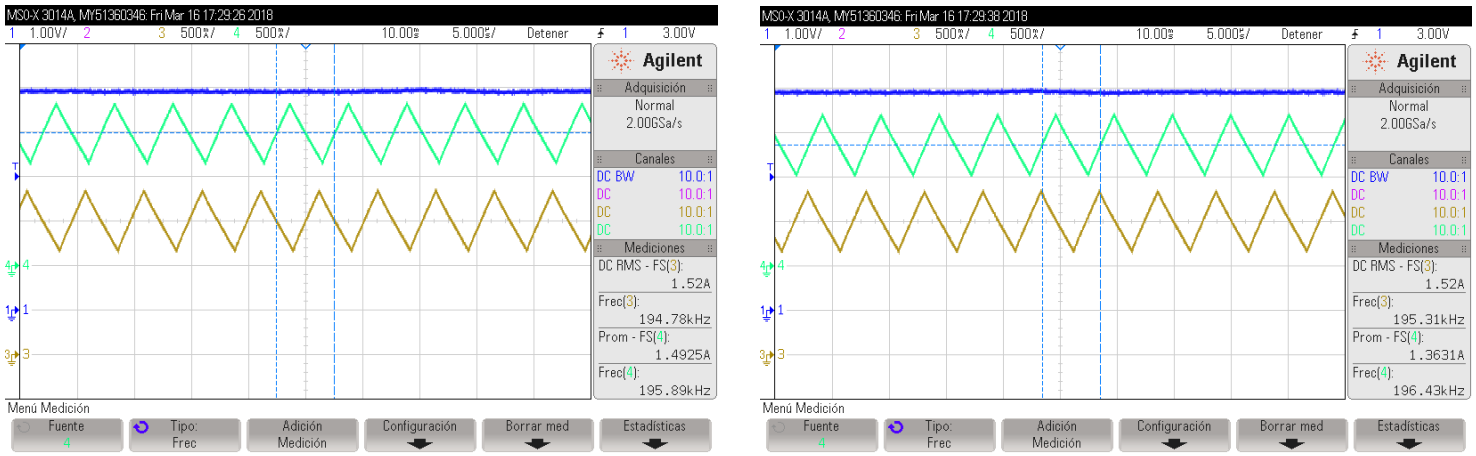


Figura 4-21: Desfase entre corrientes: fase 1/fase 3 (izda) y fase 1/fase 4 (dcha), resolución 20 bits carga 0,625 Ω

5 Conclusiones y trabajo futuro

5.1 Conclusiones

En este Trabajo de Fin de Grado se ha profundizado en el control digital de convertidores conmutados. Para ello se ha empleado una FPGA de Altera de la familia MAX10, que cuenta con ADC integrado y que, dado que se trata de una tecnología relativamente reciente, se ha tenido que estudiar cuidadosamente la configuración de la misma para lograr el funcionamiento deseado.

Una vez conocidas las características de la FPGA se ha procedido a escoger el convertidor conmutado a regular. La elección de este tipo de convertidor se ha debido fundamentalmente a su mayor eficiencia respecto a otros modelos como el lineal, así como por lo común que suele ser regular de forma digital este tipo de convertidores a día de hoy.

El convertidor conmutado escogido fue un reductor o *Buck*, en el que la tensión de salida es inferior a la de entrada y, una vez elegido este modelo, se procedió al diseño teórico del regulador mediante herramientas matemáticas como Matlab. En esta primera parte se diseñaron reguladores para las tres cargas propuestas, correspondientes a un 10% de carga para una fase, 100% de carga para una fase y cuatro fases. Durante su diseño se observó que, para lograr obtener la estabilidad en todos y cada uno de los casos, se ha de diseñar para la opción más inestable de todas y, por ello, el regulador final diseñado corresponde al modelo de 10% de carga para una fase.

Antes de pasar a su implementación en VHDL, se disminuyó la ganancia del regulador diseñado hasta que su valor inicial no sobrepasara el valor en régimen permanente. Esta modificación se realizó tras observar que con el regulador diseñado en un principio, teóricamente mucho más rápido, se obtenía una dinámica indeseable en simulación y, ya que en este trabajo ha primado la estabilidad frente a la rapidez del sistema, no supuso demasiado problema sacrificar algo de rapidez por una mayor estabilidad.

A continuación, con el regulador ya diseñado, se procedió a realizar el diseño en VHDL mediante el software propietario de Altera Quartus tras previamente haberse familiarizado con dicho programa y haber configurado en el mismo el ADC de la FPGA. En este punto se estudió la generación de las señales de PWM del reductor tanto para el caso de una fase como para el caso de cuatro fases con los disparos alternados. Además, se incidió en la configuración temporal o *timing* de las señales del regulador en cada ciclo de reloj.

Ya con todos los módulos programados, instanciados e interconectados se llevó a cabo un estudio sobre el impacto de la resolución de los coeficientes del regulador en el comportamiento global del sistema. Tras realizar múltiples pruebas y obtener gran cantidad de resultados mediante simulaciones, se llegó a la conclusión de que dicha resolución es vital para obtener la dinámica deseada. A partir de una determinada resolución los resultados obtenidos convergen a los teóricos y podría decirse que dar con esta resolución es uno de los mayores retos del diseño de este tipo de sistemas. Mientras que una menor resolución produce un comportamiento totalmente indeseable, aumentar más de lo necesario dicha resolución puede suponer un mayor tiempo de procesamiento y malgastar recursos, dependiendo de la aplicación.

Por último, una vez realizado el montaje del circuito, se pasó a tomar una serie de medidas con el objetivo de llevar a cabo una comparativa entre los resultados de simulación y los obtenidos en hardware. Se pudo observar que estos resultados diferían ligeramente debido a la diferencia de modelado entre las versiones software y hardware, pero por lo general eran bastante similares. Además, se verificó la importancia de la resolución de los coeficientes al probar distintas versiones con mayor o menor resolución a la FPGA. Para el caso de cuatro fases además se observó tanto en la simulación como en el montaje los desfases de 90º existentes entre cada fase necesarios para lograr a la salida del convertidor reductor una corriente lineal mediante la compensación de los rizados de las corrientes, así como la diferencia entre los valores de las corrientes para distintas cargas con valores superiores a menor resistencia de carga para compensar la misma.

5.2 Trabajo Futuro

Como se ha explicado a lo largo de este trabajo, la resolución escogida para los coeficientes del regulador diseñado a la hora de su digitalización es un tema fundamental en el campo del control en lazo cerrado para lograr en el sistema real la misma dinámica que en el teórico. Por ello, se podría tratar de realizar un análisis más riguroso y detallado de este tema teniendo en cuenta todos los factores influyentes e intentando obtener un modelo matemático que detalle los resultados esperados para cada resolución.

Otro punto de interés sería diseñar un sistema que permita que, de algún modo, los coeficientes se ajusten de forma automática a la resolución más conveniente o incluso realizar de nuevo el diseño empleando una representación en coma flotante, frente a la de coma fija empleada en este trabajo con el objetivo de comparar las prestaciones en ambos casos.

Además, se podría tratar de controlar otro tipo de convertidor conmutado como un *Boost* para comprobar si el impacto de la resolución es tan grande para los coeficientes de los reguladores diseñados en ese caso.

Por otro lado, se podría intentar diseñar un regulador para el caso del *Buck* que permita una mayor rapidez sin caer en la inestabilidad.

Por último, tendría sentido explorar el campo del control de sistemas en lazo cerrado mediante este tipo de FPGAs con ADC integrado ya que, pese a su aparente dificultad de configuración, ofrece unos resultados muy prometedores con vistas al futuro.

Referencias

[1] K. Morris, “Max 10 kills the CPLD,” 2014. [Online] https://www.eejournal.com/article/20140930-max10.

[2] “Intel MAX 10 FPGA Device Datasheet”, Datasheet de la FPGA MAX 10, Intel 2017

[3] Field Programmable Array (FPGA) Intel/Altera 10M50DAF484C7G Product Detail, Mouser Electronics, Inc.
https://www.mouser.es/ProductDetail/IntelAltera/10M50DAF484C7G?qs=G%2fX6g08h60vMuz000D33Dg==

[4] “DE10-Lite Schematic”, Esquemático de la placa DE10-Lite, Terasic, 2017

[5] “Intel MAX 10 Analog to Digital Converter User Guide”, Guía de usuario del ADC de la FPGA MAX 10, Intel, 2017

[6] Robert W. Erickson, Dragan Maksimovic, “Fundamentals of Power Electronics”, Norwell, Massachusetts, Kluwer Academic, 2001

[7] Laura Usero Puig, “Diseño, implementación y control de un reductor multifase”, Trabajo Fin de Grado, Universidad Autónoma de Madrid (UAM), Escuela Politécnica Superior (EPS), 2014

[8] A.V. Oppenheim, R.W. Schafer, “Discrete-Time Signal Processing”, Upper Sadddle River, NJ: Prentice Hall, 1998, pp. 354-363.

[9] P. Zumel, C. Fernández, M. Sanz, A. Lázaro, A. Barrado, “Step-by-step design of an FPGA-based digital compensator for DC/DC converters oriented to an introductory course”, IEEE Trans. on Education, vol.54, no.4, pp. 599–609, Nov. 2011.

[10] L. Corradini, D. Maksimovic, P. Mattavelli, R. Zane, “Digital control of high-frequency switched-mode power converters”, Hoboken, NJ: Wiley-IEEE Press, 2015, pp. 191-239.

[11] “DE10-Lite User Manual”, Manual de usuario de la placa Terasic, 2017

Glosario

ADC	Analog-to-Digital Converter
CC	Corriente Continua
DSP	Digital Signal Processor
EPS	Escuela Politécnica Superior
FPGA	Field-Programmable Gate Array
GPIO	General Purpose Input/Output
HCTLab	Hardware & Control Technology Laboratory
HSM	High-Side MOSFET
LSM	Low-Side MOSFET
PDI	Proportional Integral Derivative
PLL	Phase-Locked Loop
PWM	Pulse-Width Modulation
SDRAM	Synchronous Dynamic Random-Access Memory
SMD	Surface Mounted Device
TFG	Trabajo Fin de Grado
UAM	Universidad Autónoma de Madrid
USB	Universal Serial Bus
VGA	Video Graphics Array
VHDL	VHSIC Hardware Description Language

Anexos

A Diseño para otros reguladores

Se incluye a continuación el diseño realizado de otros reguladores para las demás cargas:

Diseño para carga nominal para cuatro fases

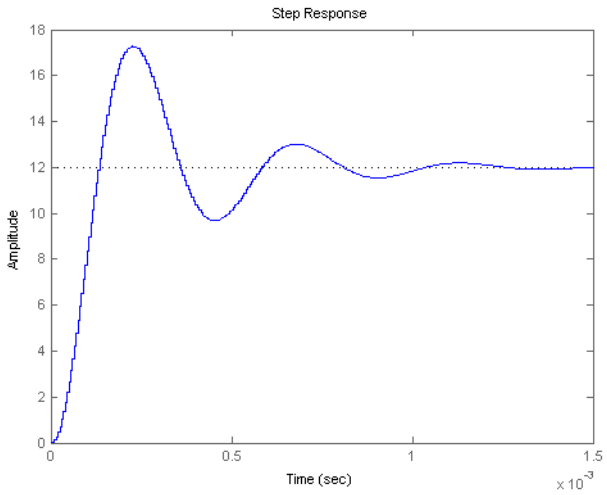
Para el caso en el que R vale 0,625 Ω (carga nominal para cuatro fases) la función de transferencia queda de la siguiente forma:

$$G_1(s) = \frac{12}{4{,}84 \cdot 10^{-9}\, s^2 + 3{,}52 \cdot 10^{-5}\, s + 1}$$

Que, haciendo la conversión entre coeficientes s y z, queda:

$$G_1(z) = \frac{0{,}03061\, z + 0{,}03024}{z^2 - 1{,}959\, z + 0{,}9643}$$

A continuación, se ha procedido a representar su respuesta al escalón mediante el programa Matlab:



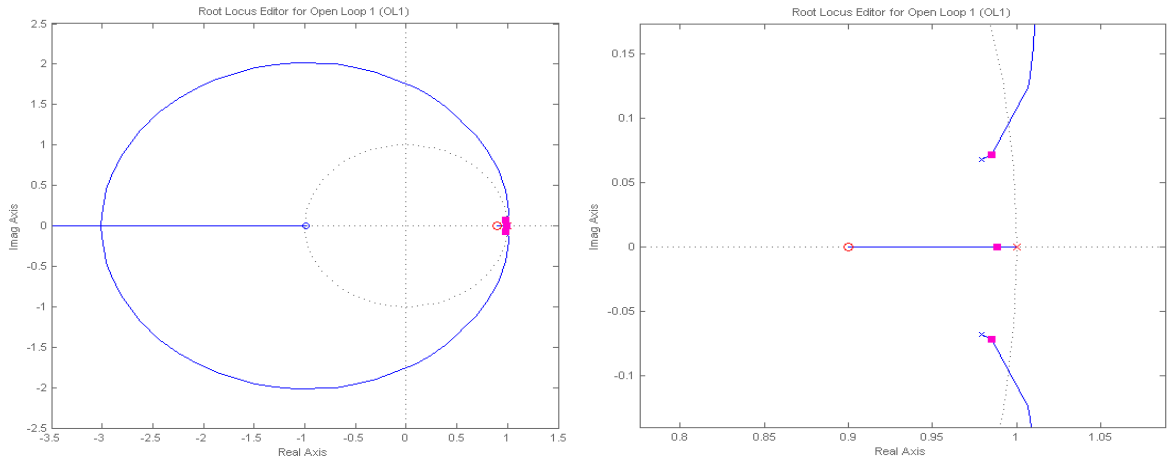
La dinámica es claramente la de un segundo orden. Se aprecian pequeñas sobreoscilaciones, pero en general, es un sistema lejano a la inestabilidad.

Se ha procedido, pues, al diseño del regulador mediante la herramienta Sisotool. Para ello, se ha tenido en cuenta que el sistema presenta dos polos complejos muy cercanos a 1 y un cero en prácticamente -1.
A la vista de esto se ha añadido un cero en 0,9 (cercano a 1) y un integrador (polo en 1).
Para lograr la estabilidad y un tiempo de establecimiento razonable se ha empleado además una ganancia pequeña, de 0,01.

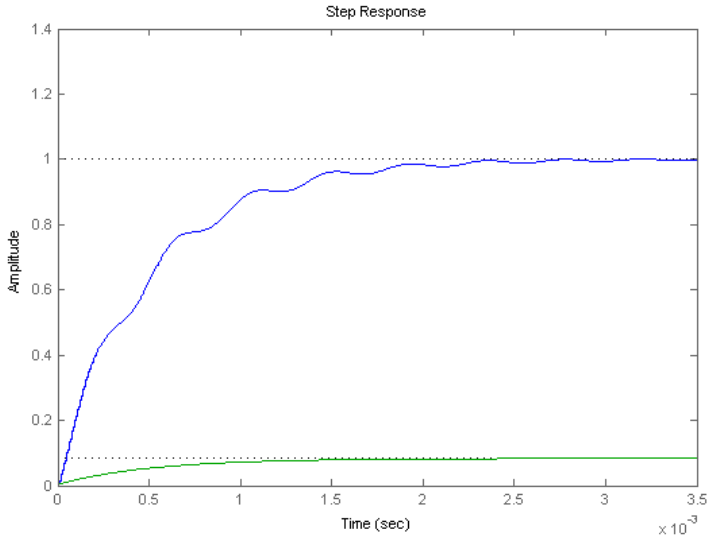
El regulador diseñado para este caso queda entonces:

$$R_1(z) = 0{,}01\frac{(z-0{,}9)}{(z-1)}$$

La posición de los polos, ceros y el lugar de las raíces se puede ver en estas figuras:



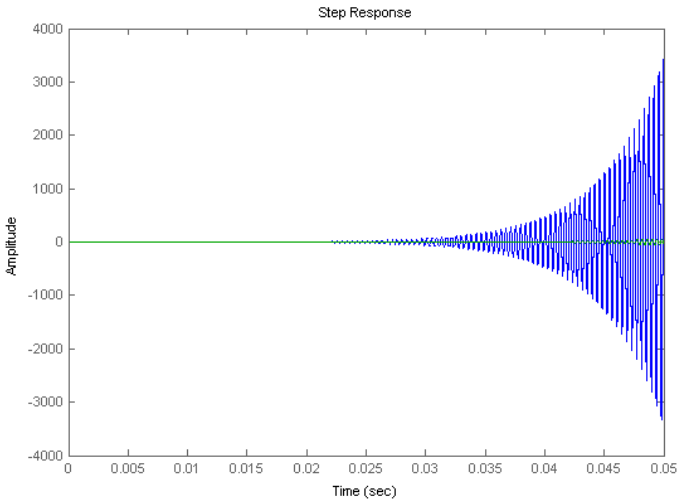
Si además se representa la actuación (verde) y la salida en lazo cerrado (azul) del sistema:



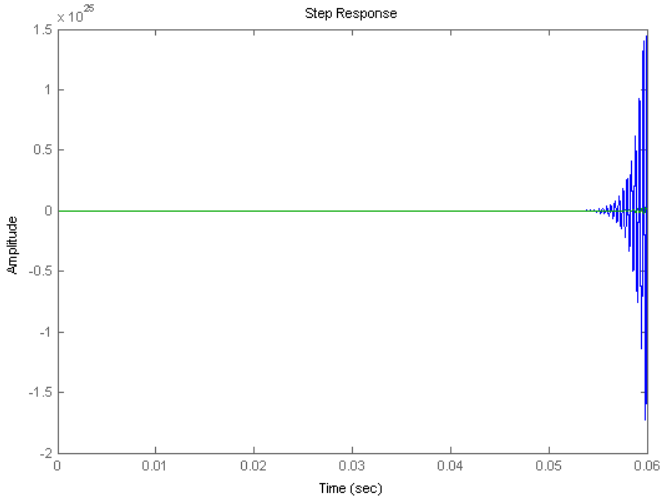
Se puede observar cómo el sistema obtenido no sobreoscila en absoluto y su dinámica es prácticamente la de un primer orden. Además, si se observa su tiempo de establecimiento (elegido con el criterio de un 5%), se puede ver que es de t_s = 1,33 ms, que se corresponde con un sistema bastante rápido.

El problema de esta aproximación viene cuando se trata de controlar con este regulador los otros dos sistemas correspondientes a los otros dos valores de R (2,5 Ω y 25 Ω).

Si se intenta con el correspondiente a carga nominal para una fase (2,5 Ω), se obtiene la siguiente salida en lazo cerrado, totalmente inestable:



Algo similar sucede si se intenta regular el correspondiente a 10% de carga para una fase:



Estos resultados dan credibilidad al principio de diseño que se ha tomado: diseñar el regulador para el caso más inestable para que, de este modo, valga también en todos los demás casos.

Diseño para carga nominal para una fase

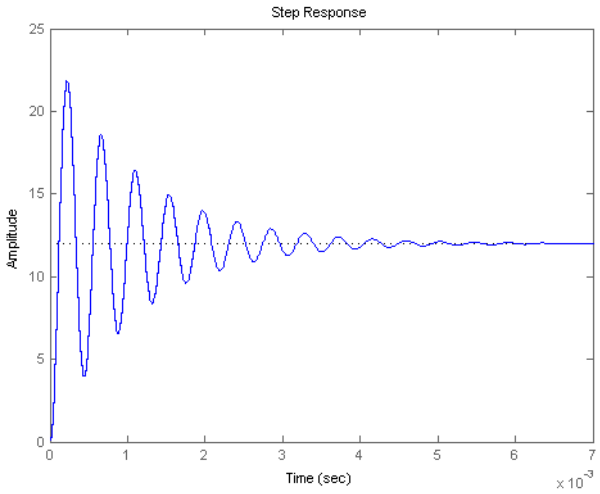
En este caso la función de transferencia, con R = 2,5 Ω, queda de la siguiente forma:

$$G_2(s) = \frac{12}{4{,}84 \cdot 10^{-9}\, s^2 + 8{,}8 \cdot 10^{-6}\, s + 1}$$

Que, en el dominio z, queda:

$$G_2(z) = \frac{0{,}03088\, z + 0{,}03079}{z^2 - 1{,}986 z + 0{,}991}$$

Cuya respuesta al escalón es la siguiente:



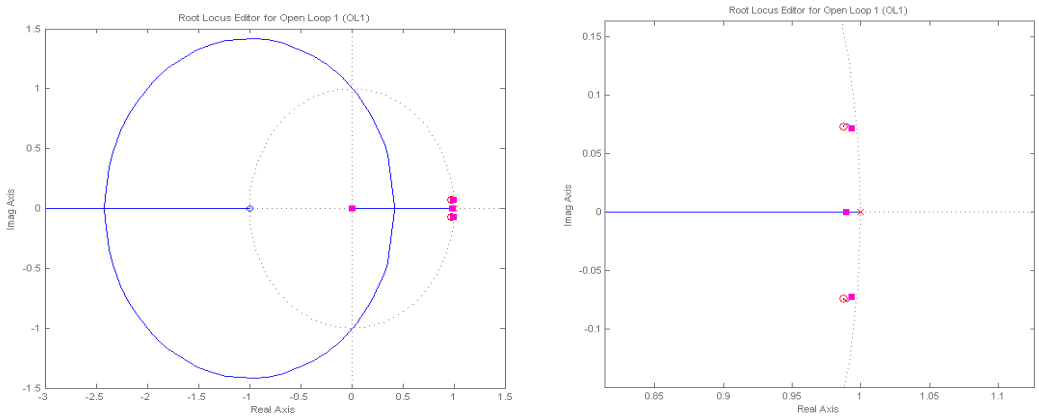
Se puede apreciar que en este caso hay una mayor sobreoscilación y, en general, menos estabilidad que en el caso anterior.

Para diseñar el regulador se ha partido de que, si bien se tiene prácticamente el mismo cero en -1 que en el caso anterior, esta vez los polos se encuentran aún más cerca de 1.
Por ello, en este caso se ha optado por añadir dos ceros complejos conjugados extremadamente cercanos a dichos polos (en 0,98 ± j 0,07), prácticamente encima de los mismos para que no causen inestabilidad. Además, se ha situado nuevamente un integrador (polo en 1) y un polo en 0 (necesario en este caso).
La ganancia, por la que en este caso se ha optado para alcanzar un compromiso entre estabilidad y rapidez, ha sido de 0,16.

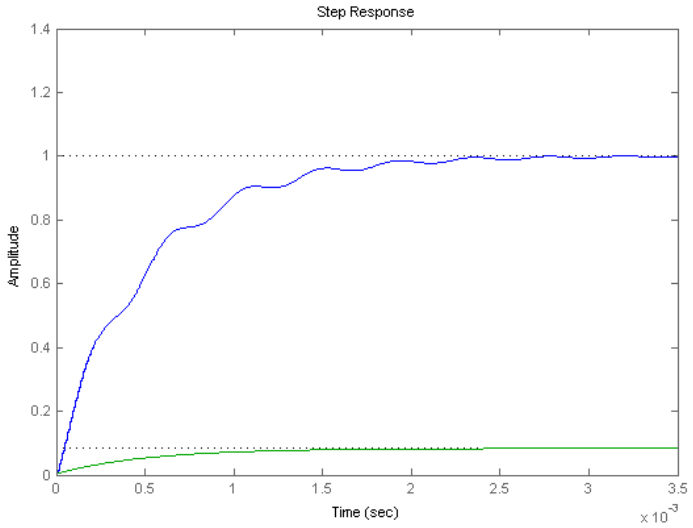
El regulador diseñado para este caso queda entonces:

$$R_2(z) = 0{,}16 \frac{(z^2 - 1{,}96 z + 0{,}965)}{z\,(z-1)}$$

La posición de los polos y ceros anteriormente descrita se puede observar con mayor detalle en las siguientes gráficas:

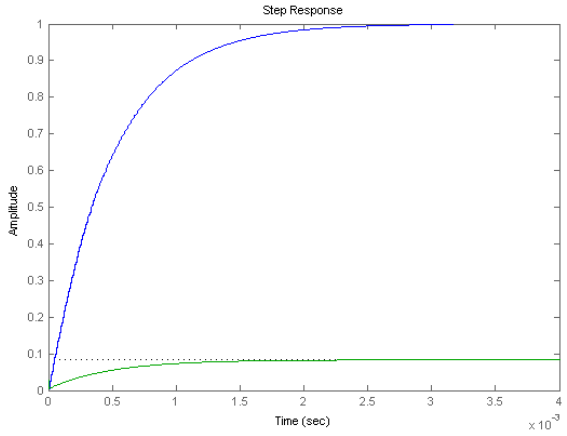


Se puede además apreciar la actuación (verde) y la salida en lazo cerrado (azul):

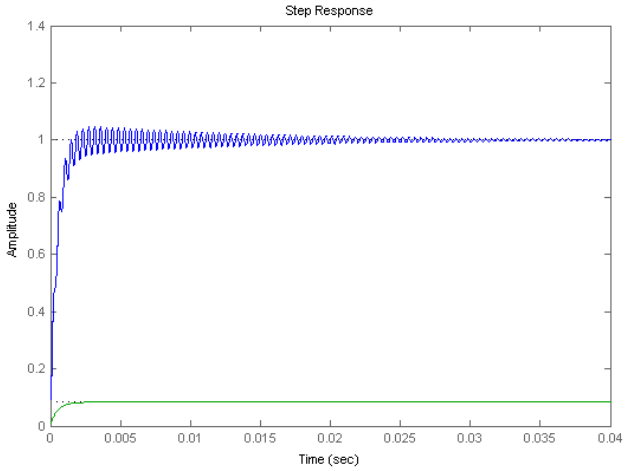


Nuevamente se ha obtenido un sistema con una dinámica muy similar a un primer orden y con un tiempo de establecimiento bastante pequeño. En este caso es de 1,43 ms.

Lo que sucede en este caso es que este regulador es además capaz de controlar sin problemas el sistema en el caso de R = 0,625 Ω, con un t_s = 1,45 ms.



Si bien dichos resultados son bastante prometedores de cara a la implementación del regulador, nuevamente sucede que no es posible controlar con este regulador el sistema correspondiente al mayor valor de resistencia (R = 25 Ω), es decir, 10% de carga para una fase, sin obtener muchas sobreoscilaciones indeseadas y un tiempo de establecimiento relativamente alto (t_s = 2,98 ms).



Por todo ello, tras realizar este estudio, se ha llegado a la conclusión de que lo más razonable es realizar el diseño para el caso de mayor inestabilidad.

B Manual del programador

Se muestra a continuación la implementación realizada en VHDL de los diferentes módulos que conforman el circuito para el caso de cuatro fases para 20 bits de resolución en su versión para bajar a placa:

GestorADC.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;


entity GestorADC is port (

   Clk      :  in std_logic;
   Reset    :  in std_logic;
   ADCout   :  out std_logic_vector(11 downto 0)

);
end GestorADC;


architecture behavioural of GestorADC is


   component ADC port (

      clk_clk                           : in  std_logic                     := '0';             --                    clk.clk
      clock_bridge_0_out_clk_clk        : out std_logic;                                        -- clock_bridge_0_out_clk.clk
      modular_adc_0_command_valid       : in  std_logic                     := '0';             --  modular_adc_0_command.valid
      modular_adc_0_command_channel     : in  std_logic_vector(4 downto 0)  := (others => '0'); --                        .channel
      modular_adc_0_command_startofpacket : in  std_logic                   := '0';             --                        .startofpacket
      modular_adc_0_command_endofpacket : in  std_logic                     := '0';             --                        .endofpacket
      modular_adc_0_command_ready       : out std_logic;                                        --                        .ready
      modular_adc_0_response_valid      : out std_logic;                                        -- modular_adc_0_response.valid
      modular_adc_0_response_channel    : out std_logic_vector(4 downto 0);                     --                        .channel
      modular_adc_0_response_data       : out std_logic_vector(11 downto 0);                    --                        .data
      modular_adc_0_response_startofpacket : out std_logic;                                     --                        .startofpacket
      modular_adc_0_response_endofpacket : out std_logic;                                       --                        .endofpacket
      reset_reset_n                     : in  std_logic                     := '0'              --                  reset.reset_n

   );
   end component;


   signal sys_clk                :  std_logic;

   signal command_valid          :  std_logic;
   signal command_channel        :  std_logic_vector(4 downto 0);
   signal command_startofpacket  :  std_logic;
   signal command_endofpacket    :  std_logic;
   signal command_ready          :  std_logic;

   signal response_valid         :  std_logic;
   signal response_channel       :  std_logic_vector(4 downto 0);
   signal response_data          :  std_logic_vector(11 downto 0);
   signal response_startofpacket :  std_logic;
   signal response_endofpacket   :  std_logic;

   signal cur_adc_ch             :  std_logic_vector(4 downto 0);
   signal adc_sample_data        :  std_logic_vector(11 downto 0);


   begin

      u0 :  ADC    port map (

         clk_clk =>  clk,
         clock_bridge_0_out_clk_clk =>    sys_clk,
         reset_reset_n =>  Reset,                                                    --Resetea a 0 (Activo bajo)
         modular_adc_0_command_valid => command_valid,
         modular_adc_0_command_channel => command_channel,
         modular_adc_0_command_startofpacket =>    command_startofpacket,
         modular_adc_0_command_endofpacket =>   command_endofpacket,
         modular_adc_0_command_ready =>   command_ready,
         modular_adc_0_response_valid =>  response_valid,
         modular_adc_0_response_channel =>   response_channel,
         modular_adc_0_response_data =>   response_data,
         modular_adc_0_response_startofpacket =>   response_startofpacket,
         modular_adc_0_response_endofpacket =>  response_endofpacket

      );

      --command_startofpacket <= '1';
      --command_endofpacket <= '1';
      command_valid <= '1';
      command_channel <= "00001";

      process (sys_clk)

      begin

         if sys_clk'event and sys_clk = '1'  then

            if response_valid = '1' then

               adc_sample_data <= response_data;
               cur_adc_ch <= response_channel;

            end if;

         end if;

      end process;

      ADCout <= adc_sample_data;


end behavioural;
```

Sincro_PWM.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;
use ieee.fixed_float_types.all;                -- ieee_proposed for VHDL-93 version
use ieee.fixed_pkg.all;


entity Sincro_PWM is port (

   Clk               : in  std_logic;
   Reset             : in  std_logic;

   Consigna          : in  std_logic_vector (11 downto 0);
   CapturaConsigna   : in  std_logic;

   ADCout            : in  std_logic_vector (11 downto 0);
   Actuacion         : in  sfixed (12 downto 0);   --Actuacion

   Swap              : in std_logic;

   HSM_1             : out    std_logic;                          --Primera Fase
   LSM_1             : out    std_logic;

   HSM_2             : out    std_logic;                          --Segunda Fase
   LSM_2             : out    std_logic;

   HSM_3             : out    std_logic;                          --Tercera Fase
   LSM_3             : out    std_logic;

   HSM_4             : out    std_logic;                          --Cuarta Fase
   LSM_4             : out    std_logic;

   CE                : out std_logic;

   error             : out sfixed(4 downto -8)

);
end Sincro_PWM;


architecture behavioural of Sincro_PWM is

   signal contador_1             :  unsigned (7 downto 0) := "00000000";
   signal contador_2             :  unsigned (7 downto 0) := "00000000";
   signal contador_3             :  unsigned (7 downto 0) := "00000000";
   signal contador_4             :  unsigned (7 downto 0) := "00000000";

   signal CicloTrabajo           :  std_logic_vector (7 downto 0);
   signal CicloMasDT             :  std_logic_vector (7 downto 0);

   signal ConsignaRegistrada     :  std_logic_vector (11 downto 0);

   signal errorint               :  std_logic_vector (12 downto 0);

   constant DT                   :  std_logic_vector (7 downto 0) := "00000001";
   constant MaxContDT            :  std_logic_vector (7 downto 0) := "11111110";


   begin

      --    PROCESO DEL CONTADOR DE CICLOS    --
      ----------------------------------------------
      ----------------------------------------------

      process (Clk,Reset)

      begin

         if Reset = '0' then                              -- Si el reset está activo la salida vale 0

            contador_1 <= (others => '0');
            contador_2 <= "01000000";
            contador_3 <= "10000000";
            contador_4 <= "11000000";

         elsif rising_edge (Clk) then                     -- Si hay un flanco de subida del reloj

            contador_1 <= contador_1 + 1;--"00000001";           -- Contador principal (PRIMERA FASE)
            contador_2 <= contador_2 + 1;--"01000000";           -- Contador principal + 64 (SEGUNDA FASE)
            contador_3 <= contador_3 + 1;--"10000000";           -- Contador principal + 128 (TERCERA FASE)
            contador_4 <= contador_4 + 1;--"11000000";           -- Contador principal + 192 (CUARTA FASE)

         end if;

      end process;


      --   CONDICIÓN PARA ACTIVACIÓN DE "CE"  --
      ----------------------------------------------
      -- "CE" se activa en el ciclo N-4 (252) --

      process (Clk,Reset)

      begin

         if Reset = '0' then

            CE <= '0';

         elsif rising_edge (Clk) then

            if contador_1 = "11111011" then

               CE <= '1';

            else

               CE <= '0';

            end if;

         end if;

      end process;
```

```
--          GESTIÓN DEL "PWM"          --
-----------------------------------------
-- PWM A 1 DURANTE EL CICLO DE TRABAJO --
process (Clk,Reset)
begin
   if Reset = '0' then
      HSM_1 <= '0';
      LSM_1 <= '0';
      HSM_2 <= '0';
      LSM_2 <= '0';
      HSM_3 <= '0';
      LSM_3 <= '0';
      HSM_4 <= '0';
      LSM_4 <= '0';
   elsif rising_edge (Clk) then
      --FASE 1
      if contador_1 < unsigned (CicloTrabajo) then -- Pwm se mantiene a 1 mientras que contador sea menor que el ciclo de trabajo
         HSM_1 <= '1';
      else
         HSM_1 <= '0';
      end if;
      if (contador_1 > unsigned (CicloMasDT)) and (contador_1 < unsigned(MaxContDT)) then
         LSM_1 <= '1';
      else
         LSM_1 <= '0';
      end if;
      --FASE 2
      if contador_2 < unsigned (CicloTrabajo) then -- Pwm se mantiene a 1 mientras que contador sea menor que el ciclo de trabajo
         HSM_2 <= '1';
      else
         HSM_2 <= '0';
      end if;
      if (contador_2 > unsigned (CicloMasDT)) and (contador_2 < unsigned(MaxContDT)) then
         LSM_2 <= '1';
      else
         LSM_2 <= '0';
      end if;
      --FASE 3
      if contador_3 < unsigned (CicloTrabajo) then -- Pwm se mantiene a 1 mientras que contador sea menor que el ciclo de trabajo
         HSM_3 <= '1';
      else
         HSM_3 <= '0';
      end if;
      if (contador_3 > unsigned (CicloMasDT)) and (contador_3 < unsigned(MaxContDT)) then
         LSM_3 <= '1';
      else
         LSM_3 <= '0';
      end if;
      --FASE 4
      if contador_4 < unsigned (CicloTrabajo) then -- Pwm se mantiene a 1 mientras que contador sea menor que el ciclo de trabajo
         HSM_4 <= '1';
      else
         HSM_4 <= '0';
      end if;
      if (contador_4 > unsigned (CicloMasDT)) and (contador_4 < unsigned(MaxContDT)) then
         LSM_4 <= '1';
      else
         LSM_4 <= '0';
      end if;
   end if;
end process;
```

```
        --       CAPTURA DE LA CONSIGNA        --
        ----------------------------------------
        -- PWM A 1 DURANTE EL CICLO DE TRABAJO --
        process (Clk,Reset)
        begin
           if Reset = '0' then
              ConsignaRegistrada  <= (others => '0');
           elsif rising_edge (Clk) then
              if (CapturaConsigna = '0' ) then
                 ConsignaRegistrada <= Consigna;
              end if;
           end if;
        end process;


        -- CÁLCULO DEL ERROR --
        errorint <= std_logic_vector(signed('0' & ConsignaRegistrada) - (signed('0' & ADCout)));
        error <= to_sfixed(errorint, 4 , -8);



        CicloTrabajo <=   to_slv(Actuacion(11 downto 4)) when Swap = '0' else
                          Consigna(11 downto 4);
        CicloMasDT   <=   CicloTrabajo + DT;

    end behavioural;
```

Regulador.vhd

```
library work, ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.fixed_float_types.all;                 -- ieee_proposed for VHDL-93 version
use ieee.fixed_pkg.all;


entity Regulador is port (

   Clk         : in     std_logic;
   Reset       : in     std_logic;
   CE          : in     std_logic;

   Error       : in     sfixed (4 downto -8);
   Actuacion   : out    sfixed (12 downto 0)
);
end Regulador;


architecture behavioural of Regulador is

   --entrada y salida
   signal sX : sfixed (4 downto -8);
   signal sY : sfixed (0 downto -12);

   --valores anteriores de la entrada
   signal sX0 : sfixed (4 downto -8);
   signal sX1 : sfixed (4 downto -8);
   signal sX2 : sfixed (4 downto -8);

   --coeficientes
   signal B0 : sfixed (0 downto -19) := to_sfixed(0.08, 0 , -19);
   signal B1 : sfixed (0 downto -19) := to_sfixed(-0.1552, 0, -19);
   signal B2 : sfixed (0 downto -19) := to_sfixed(0.075272, 0, -19);

   signal A1 : sfixed (1 downto 0)  := to_sfixed(-1, 1, 0);

   --resultados parciales
   signal sB0X0 : sfixed (5 downto -27);
   signal sB1X1 : sfixed (5 downto -27);
   signal sB2X2 : sfixed (5 downto -27);


   signal sA1Y1 : sfixed (8 downto -27);

   --resultados parciales REGISTRADOS
   signal sB0X0Reg : sfixed (5 downto -27);
   signal sB1X1Reg : sfixed (5 downto -27);
   signal sB2X2Reg : sfixed (5 downto -27);

   signal sA1Y1Reg : sfixed (8 downto -27);

   -- salida en máxima resolucion
   signal sY_6_27 : sfixed (6 downto -27);

   signal sY_6_27int : sfixed (6 downto -27);

   -- salida en máxima resolucion REGISTRADA
   signal sY_6_27Reg : sfixed (6 downto -27);

   -- guardar salida anterior en máxima resolución
   signal sY1 : sfixed (6 downto -27);
   signal sY2 : sfixed (6 downto -27);

   signal Actuacionc : std_logic_vector (12 downto 0);

   constant limite_superior : sfixed (6 downto -27) := to_sfixed(0.95, 6 , -27);
   constant limite_inferior : sfixed (6 downto -27) := to_sfixed (0.0, 6, -27);

   begin

   --*********************************************************************--
   --                --------(0.08-0.1552z^-1+0.075272z^-2)---------     --
   --               R(z) =  --------------------------             --
   --                --------         (1-z^-1)          ---------     --
   --*********************************************************************--


   registro1: process (Clk, Reset)                      -- El "paso del tiempo" debe ser síncrono

   begin

      if (Reset = '0') then                             -- Reset de elementos internos.

         sX0 <= (others => '0');
         sX1 <= (others => '0');
         sX2 <= (others => '0');

         sY1 <= (others => '0');
         sY2 <= (others => '0');

      elsif (rising_edge(Clk)) then

         if (CE = '1') then                             -- La actualización debe ser síncrona

            sX0 <= sX;
            sX1 <= sX0;
            sX2 <= sX1;

            sY1 <= sY_6_27Reg;
            sY2 <= sY1;

         end if;

      end if;

   end process;

   -- Conversiones entre la gestión de datos interna y el exterior

   -- Entrada
   sX <= Error;

   -- Salida
   Actuacionc <= to_slv(sY);
   Actuacion <= to_sfixed(Actuacionc,12,0);


   -- Cálculo de las operaciones intermedias
   sB0X0 <= resize(sX0 * B0, sB0X0);
   sB1X1 <= resize(sX1 * B1, sB1X1);
   sB2X2 <= resize(sX2 * B2, sB2X2);

   sA1Y1 <= resize(sY1 * A1, sA1Y1);
```

```
--REGISTRAMOS sB0X0, sB1X1, sB2X2 y sA1Y1
registro2: process (Clk, Reset)
begin
   if (Reset = '0') then
      sB0X0Reg <= (others => '0');
      sB1X1Reg <= (others => '0');
      sB2X2Reg <= (others => '0');
      sA1Y1Reg <= (others => '0');
   elsif (rising_edge(Clk)) then
      sB0X0Reg <= sB0X0;
      sB1X1Reg <= sB1X1;
      sB2X2Reg <= sB2X2;
      sA1Y1Reg <= sA1Y1;
   end if;
end process;


-- Calcular el valor de la salida actual
sY_6_27 <= resize (sB0X0Reg + sB1X1Reg + sB2X2Reg - sA1Y1Reg, sY_6_27); -- Mantener gran resolución para no perder información

--Limitamos sY_6_27
sY_6_27int <=  (others => '0')       when sY_6_27 < limite_inferior else
               limite_superior       when sY_6_27 > limite_superior else
               sY_6_27;

  --REGISTRAMOS sY_6_27
  registro3: process (Clk, Reset)
  begin
     if (Reset = '0') then
        sY_6_27Reg <= (others => '0');
     elsif (rising_edge(Clk)) then
        sY_6_27Reg <= sY_6_27int;
     end if;

  end process;

  -- Transformar el valor al tamaño de salida
  sY <= resize (sY_6_27Reg,sY);

end behavioural;
```

Controlador.vhd (Top Level)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.fixed_float_types.all;                         -- ieee_proposed for VHDL-93 version
use ieee.fixed_pkg.all;

entity Controlador is port (

    Clk                 : in    std_logic;                                  --Reloj del sistema
    Reset               : in    std_logic;                                  --SeÃ±al de Reset

    Swap                : in    std_logic;                                  --Selector de valor para PWM

    Consigna            : in    std_logic_vector (11 downto 0);         --Consigna del lazo

    CapturaConsigna     : in    std_logic;                                  --Captura de Consigna


    HSM_1               : out   std_logic;                                  --Primera Fase
    LSM_1               : out   std_logic;

    HSM_2               : out   std_logic;                                  --Segunda Fase
    LSM_2               : out   std_logic;

    HSM_3               : out   std_logic;                                  --Tercera Fase
    LSM_3               : out   std_logic;

    HSM_4               : out   std_logic;                                  --Cuarta Fase
    LSM_4               : out   std_logic;

    LEDs                : out       std_logic_vector (9 downto 0)           --Salida a los LEDs (mostraremos el valor del ADC)

);
end Controlador;


architecture behavioural of Controlador is


    -- Gestor ADC
    component GestorADC port(

        Clk     :   in  std_logic;
        Reset   :   in  std_logic;
        ADCout  :   out std_logic_vector(11 downto 0)

    );
    end component;

   -- Gestor PWM
    component Sincro_PWM port(

       Clk                  : in  std_logic;
       Reset                : in  std_logic;

       Consigna             : in    std_logic_vector (11 downto 0);
       CapturaConsigna      : in    std_logic;

       ADCout               : in    std_logic_vector (11 downto 0);
       Actuacion            : in    sfixed (12 downto 0);   --Actuacion

       Swap                 : in std_logic;

       HSM_1                : out std_logic;
       LSM_1                : out std_logic;

       HSM_2                : out std_logic;
       LSM_2                : out std_logic;

       HSM_3                : out std_logic;
       LSM_3                : out std_logic;

       HSM_4                : out std_logic;
       LSM_4                : out std_logic;

       CE                   : out std_logic;

       error                : out sfixed(4 downto -8)

   );
   end component;


   -- Regulador
   component  Regulador port (

       Clk          : in    std_logic;
       Reset        : in    std_logic;
       CE           : in    std_logic;
       Error        : in    sfixed (4 downto -8);
       Actuacion    : out   sfixed (12 downto 0)

   );
   end component;


   -- SeÃ±ales internas
   signal ADC_PWM       :   std_logic_vector (11 downto 0);
   signal CEc           :   std_logic;
   signal errorc        :   sfixed (4 downto -8);
   signal Actuacionc    :   sfixed (12 downto 0);
```

```
begin

-- Instanciacion Gestor ADC
    Inst_GestorADC: GestorADC port map(

        Clk => Clk,
        Reset => Reset,
        ADCout => ADC_PWM

    );

-- Instanciacion Gestor PWM
Inst_Sincro_PWM: Sincro_PWM port map(

    Clk => Clk,
    Reset => Reset,

    Consigna => Consigna,
    CapturaConsigna => CapturaConsigna,

    ADCout => ADC_PWM,
    Actuacion => Actuacionc,

    Swap => Swap,

    HSM_1 => HSM_1,
    LSM_1 => LSM_1,

    HSM_2 => HSM_2,
    LSM_2 => LSM_2,

    HSM_3 => HSM_3,
    LSM_3 => LSM_3,

    HSM_4 => HSM_4,
    LSM_4 => LSM_4,

    CE => CEc,

    error => errorc

);

-- Instanciar el Regulador
Inst_Regulador: Regulador port map(

    Clk => Clk,
    Reset => Reset,
    CE => CEc,
    Error => errorc,
    Actuacion => Actuacionc

 );

  LEDs <= ADC_PWM (11 downto 2);

end behavioural;
```

El modelo del *Buck* empleado para cuatro fases:

BuckMultifaseReal.vhd

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
use IEEE.math_real.all;

   entity BuckMultifaseReal is
      port (
         --In
         Clk : in std_logic;
         Reset : in std_logic;
         HSM_1 : in std_logic;
         LSM_1 : in std_logic;
         HSM_2 : in std_logic;
         LSM_2 : in std_logic;
         HSM_3 : in std_logic;
         LSM_3 : in std_logic;
         HSM_4 : in std_logic;
         LSM_4 : in std_logic;
         Vin   : in real;
         Ir    : in real;
         --Out
         Iin   : out real;
         Vout  : out real
      );
   end BuckMultifaseReal;

architecture Reductor of BuckMultifaseReal is


   constant C        : real := 0.000220;
   constant L        : real := 0.000022;
   constant dt       : real := 0.000000020; -- 20 ns

   constant dtL      : real := dt/L;
   constant dtC      : real := dt/C;

   constant Iini     : real := 0.0;
   constant Vini     : real := 0.0;

   signal Iinaux_1, Iinaux_2, Iinaux_3, Iinaux_4 : real := 0.0;
   signal Voaux      : real := 0.0;

   signal Ic         : real := 0.0;
   signal Vl_1       : real := 0.0;
   signal Il_1       : real := 0.0;
   signal Vl_2       : real := 0.0;
   signal Il_2       : real := 0.0;
   signal Vl_3       : real := 0.0;
   signal Il_3       : real := 0.0;
   signal Vl_4       : real := 0.0;
   signal Il_4       : real := 0.0;
   signal Il_total   : real := 0.0;
begin

   Iin <= Iinaux_1+Iinaux_2+Iinaux_3+Iinaux_4;
   Il_total <= Il_1+Il_2+Il_3+Il_4;
   Ic <= Il_total - Ir;
   Vout <= Voaux;

   Interruptores : process (HSM_1,LSM_1,HSM_2,LSM_2,HSM_3,LSM_3,HSM_4,LSM_4, Vl_1, Vl_2, Vl_3, Vl_4,Vin, Voaux)
   begin

      ----------------------------------- PRIMERA FASE ---------------------------------

      if (HSM_1 = '1' and LSM_1 ='0') then

         Vl_1 <= Vin-Voaux;
         Iinaux_1 <= Il_1;

      elsif (HSM_1 = '0' and LSM_1 ='1') then

         Vl_1 <= -Voaux;
         Iinaux_1 <= 0.0;

      elsif (HSM_1 = '0' and LSM_1 ='0') then

         if Il_1<0.0 then

            Vl_1 <= Vin-Voaux;
            Iinaux_1 <= Il_1;


         else

            Vl_1 <= -Voaux;
            Iinaux_1 <= 0.0;

         end if;

      elsif (HSM_1 = '1' and LSM_1 ='1') then --Caso a evitar

         Vl_1 <= Vini;
         Iinaux_1 <= Iini;

      end if;


      ----------------------------------- SEGUNDA FASE ---------------------------------

      if (HSM_2 = '1' and LSM_2 ='0') then

         Vl_2 <= Vin-Voaux;
         Iinaux_2 <= Il_2;

      elsif (HSM_2 = '0' and LSM_2 ='1') then

         Vl_2 <= -Voaux;
         Iinaux_2 <= 0.0;

      elsif (HSM_2 = '0' and LSM_2 ='0') then

         if Il_2<0.0 then

            Vl_2 <= Vin-Voaux;
            Iinaux_2 <= Il_2;

         else

            Vl_2 <= -Voaux;
            Iinaux_2 <= 0.0;

         end if;

      elsif (HSM_2 = '1' and LSM_2 ='1') then --Caso a evitar

         Vl_2 <= Vini;
         Iinaux_2 <= Iini;

      end if;
```

```
        ---------------------------------- TERCERA FASE ----------------------------------
        if (HSM_3 = '1' and LSM_3 ='0') then

           Vl_3 <= Vin-Voaux;
           Iinaux_3 <= Il_3;

        elsif (HSM_3 = '0' and LSM_3 ='1') then

           Vl_3 <= -Voaux;
           Iinaux_3 <= 0.0;

        elsif (HSM_3 = '0' and LSM_3 ='0') then

           if Il_3<0.0 then

              Vl_3 <= Vin-Voaux;
              Iinaux_3 <= Il_3;

           else

              Vl_3 <= -Voaux;
              Iinaux_3 <= 0.0;

           end if;

        elsif (HSM_3 = '1' and LSM_3 ='1') then --Caso a evitar

           Vl_3 <= Vini;
           Iinaux_3 <= Iini;

        end if;

        ---------------------------------- CUARTA FASE ----------------------------------
        if (HSM_4 = '1' and LSM_4 ='0') then

           Vl_4 <= Vin-Voaux;
           Iinaux_4 <= Il_4;

        elsif (HSM_4 = '0' and LSM_4 ='1') then

           Vl_4 <= -Voaux;
           Iinaux_4 <= 0.0;

        elsif (HSM_4 = '0' and LSM_4 ='0') then

           if Il_4<0.0 then

              Vl_4 <= Vin-Voaux;
              Iinaux_4 <= Il_4;

           else

              Vl_4 <= -Voaux;
              Iinaux_4 <= 0.0;

           end if;

        elsif (HSM_4 = '1' and LSM_4 ='1') then --Caso a evitar

           Vl_4 <= Vini;
           Iinaux_4 <= Iini;

        end if;

     end process Interruptores;


     assigment : process (Clk, Reset)
     begin

        if Reset = '0' then

           Voaux <= Vini;
           Il_1 <= Iini;
           Il_2 <= Iini;
           Il_3 <= Iini;
           Il_4 <= Iini;

        elsif rising_edge(Clk) then

           Il_1 <= Il_1 + Vl_1*dtL;
           Il_2 <= Il_2 + Vl_2*dtL;
           Il_3 <= Il_3 + Vl_3*dtL;
           Il_4 <= Il_4 + Vl_4*dtL;
           Voaux <= Voaux + Ic*dtC;

        end if;

     end process assigment;

end Reductor;
```

Y el *testbench* diseñado para el caso de cuatro fases:

ClosedLoopBUCK_tb.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.numeric_std.all;
use ieee.math_real.all;
use ieee.std_logic_unsigned.all;


entity ClosedLoopBUCK_tb is
end ClosedLoopBUCK_tb;

architecture behavioural of ClosedLoopBUCK_tb is

   -- Instanciar el Controlador a probar
   component Controlador port (

      Clk               : in     std_logic;                          --Reloj del sistema
      Reset             : in     std_logic;                          --Señal de Reset

      Swap              : in     std_logic;                          --Selector de valor para PWM

      Consigna          : in     std_logic_vector (11 downto 0);     --Consigna del lazo

      CapturaConsigna   : in     std_logic;                          --Captura de Consigna

      ADCIn             : in     std_logic_vector (11 downto 0);     --Entrada al ADC

      HSM_1             : out    std_logic;                          --Primera Fase
      LSM_1             : out    std_logic;

      HSM_2             : out    std_logic;                          --Segunda Fase
      LSM_2             : out    std_logic;

      HSM_3             : out    std_logic;                          --Tercera Fase
      LSM_3             : out    std_logic;

      HSM_4             : out    std_logic;                          --Cuarta Fase
      LSM_4             : out    std_logic;

      LEDs              : out    std_logic_vector (9 downto 0)       --Salida a los LEDs (mostraremos el valor del ADC)

   );
   end component;

   -- Instanciar el modelo de la planta
   component BuckMultifaseReal port (

      --In
      Clk   : in std_logic;
      Reset : in std_logic;

      HSM_1 : in std_logic;
      LSM_1 : in std_logic;

      HSM_2 : in std_logic;
      LSM_2 : in std_logic;

      HSM_3 : in std_logic;
      LSM_3 : in std_logic;

      HSM_4 : in std_logic;
      LSM_4 : in std_logic;

      Vin   : in real;
      Ir    : in real;

      --Out
      Iin   : out real;
      Vout  : out real

   );
   end component;


   -- Señales
   signal clk, reset, swap : std_logic := '0';
   signal adc : std_logic_vector (11 downto 0) := (others => '0');
   signal consigna : std_logic_vector (11 downto 0) := (others => '0');
   signal capturaConsigna : std_logic := '0';
   signal ledsc : std_logic_vector (9 downto 0);

   signal HSM_1c :   std_logic;
   signal LSM_1c  :  std_logic;

   signal HSM_2c :   std_logic;
   signal LSM_2c  :  std_logic;

   signal HSM_3c :   std_logic;
   signal LSM_3c  :  std_logic;

   signal HSM_4c :   std_logic;
   signal LSM_4c  :  std_logic;

   signal Vgc  :  real;
   signal Irc  :  real;
   signal ILc  :  real;


   signal vanalogica : real;


   signal end_simc : boolean := false;

   constant Uno_entre_R :  real  := 1.0/0.625;

    -- Clock period definitions
   constant CLK_period : time := 20 ns; -- 50 MHz
```

```
    begin

       -- Mapeo de puertos del elemento a probar
    uut : Controlador port map(

       Clk => clk,
       Reset => reset,
       Swap => swap,

       ADCIn => adc,

       Consigna => consigna,
       CapturaConsigna => capturaConsigna,

       HSM_1 => HSM_1c,
       LSM_1 => LSM_1c,

       HSM_2 => HSM_2c,
       LSM_2 => LSM_2c,

       HSM_3 => HSM_3c,
       LSM_3 => LSM_3c,

       HSM_4 => HSM_4c,
       LSM_4 => LSM_4c,

       LEDs => ledsc

    );

    -- Mapeo de la planta
    planta : BuckMultifaseReal port map(

       --In

       Clk      => clk,
       Reset    => reset,

       HSM_1    => HSM_1c,
       LSM_1    => LSM_1c,

       HSM_2    => HSM_2c,
       LSM_2    => LSM_2c,

       HSM_3    => HSM_3c,
       LSM_3    => LSM_3c,

       HSM_4    => HSM_4c,
       LSM_4    => LSM_4c,

       Vin      => Vgc,
       Ir       => Irc,

      -- Out
      Iin      => ILc,
      Vout     => vanalogica

   );


   Vgc   <= 12.0;

   -- Proceso del reloj
   clk_process : process

   begin

      while not end_simc loop
         clk <= '0';
         wait for CLK_period/2;
         clk <= '1';
         wait for CLK_period/2;
      end loop;
      wait;

   end process;


   -- Proceso de simulación del ADC
   adc_proc : process

   variable tmpVanalog : std_logic_vector (11 downto 0);

   begin

      while not end_simc loop

         wait for 10 ns; --t6

         tmpVanalog := STD_LOGIC_VECTOR(CONV_UNSIGNED(INTEGER((4095.0*vanalogica)/16.0),12));   --4095=(2^12)-1 (12 bits resolución ADC)

         Irc   <= vanalogica*Uno_entre_R;

         if (vanalogica > 16.0) then
            adc <= x"FFF";
         elsif vanalogica < 0.0 then
            adc <= x"000";
         else
            adc <= tmpVanalog;
         end if;


      end loop;
   wait;

   end process;


    -- Stimulus process
   stim_proc: process

   begin

      -- hold reset state for 100 ns.
      Reset <= '0';
      Consigna <= x"000";
      wait for 100 ns;
      Reset <= '1';
      Consigna <= x"500";
      wait for 60 ms;
      Consigna <= x"600";
      wait for 60 ms;
      Consigna <= x"C00";
      wait for 60 ms;
      Consigna <= x"500";
      wait for 60 ms;
      Consigna <= x"000";
      wait for 60 ms;
      Consigna <= x"C00";
      wait for 60 ms;

      end_simc <= true;
      wait;

   end process;

end behavioural;
```